Spatio-Temporal Adaptive Algorithm for Reacting Flows

Vol 1

by

Mehtab M. Pervaiz

B.S. Mechanical Engineering, N.E.D. University, Karachi, Pakistan, 1979
M.S. Mathematics, Oklahoma University, Norman, Oklahoma, 1984

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy
in
Computational Fluid Dynamics
Department of Aeronautics and Astronautics
at the
Massachusetts Institute of Technology

May 1988

Signature of Author _____
Department of Aeronautics and Astronautics
April 29, 1988

Certified by _____
Professor Judson R. Baron
Thesis Supervisor, Department of Aeronautics and Astronautics

Certified by _____
Professor Earll M. Murman
Department of Aeronautics and Astronautics

Certified by _____
Professor Eugene E. Covert
Department of Aeronautics and Astronautics

Accepted by _____
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

SPATIO-TEMPORAL ADAPTIVE ALGORITHM FOR REACTING FLOWS

by

MEHTAB M. PERVAIZ

Submitted to the Department of Aeronautics and Astronautics on
April 29, 1988
in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy in Computational Fluid Dynamics

Abstract

Consideration is given to the numerical integration of unsteady two-dimensional flow fields which involve finite rate chemistry and are expressed in terms of conservative form Euler and species conservation equations. The coupled behavior between fluid flow and finite rate chemistry can introduce appreciable stiffness into numerical schemes, which then involve prohibitively long computation times. Such calculations become even more expensive when globally fine grid resolution, in both space and time, is utilized to ensure the capture of local flow features. However, the retention of fine grid resolution is generally needed only within small portions of the overall space/time domain. Typically, spatial resolution is desired in those regions that are characterized by steep local changes, *e.g.*, including a shock or a chemical adjustment. Similarly temporal resolution is needed both when there are non-equilibrium source terms which produce large temporal gradients, and in regions of spatially fine cells due to coupling of the time-steps with cell volumes. The aim is to provide a description of a controlled grid resolution approach in both space and time, and to demonstrate its effectiveness for a selected class of problems. An efficient spatio-temporal adaptive algorithm which allows simultaneous resolution of both temporal and spatial grids for conservation equations is presented. It is demonstrated that the approach can yield orders of magnitude faster computations at essentially the same accuracy as the globally fine grids. The algorithm uses quadrilateral cells and embedded meshes which track the moving flow features. It also allows for spatially varying time-steps which are multiples of global minimum time-steps. The adaptive technique refines the spatial and/or temporal grid whenever preselected differences exceed certain threshold levels. Results for internal flow problems are presented to demonstrate the accuracy and computational efficiency of this algorithm. Examples include blast waves and scramjet inlets. The chemistry models include a Lighthill gas and a two reaction hydrogen combustion.

Thesis Supervisor: Dr. Judson R. Baron
Title: Professor of Aeronautics and Astronautics

Acknowledgements

I am grateful to the whole M.I.T. community in providing an atmosphere that is conducive to learning and basic research. I am indebted to the faculty and my peers for providing communication channels and outlets that have helped me grow intellectually and develop a better understanding of fluid mechanics.

First and foremost, I would like to thank my chairman and advisor, Professor Judson Baron, for his guidance and encouragement and his unique way of instilling enthusiasm among his students. He has been a fatherly figure to me and has at times provided pep talks when I have been out of line and has lent a sympathetic ear to my grieviences.

I would also like to express my appreciation to other members of my committee, Prof. Earll Murman and Eugene Covert, for their helpful suggestions and discussions. In addition I would like to thank the official readers of this thesis, Prof. Michael Giles and Dr. James Elgin, for taking time out of their busy routines and providing valuable comments which were helpful in transforming the initial draft to the current form.

Thanks to Cathy Mavriplis, Dana Lindquist and John Kallinderis for reading the crude forms of the thesis on a chapter by chapter basis and providing helpful comments pertaining to both the technical and aesthetic aspects of writing. I am also grateful to John Dannenhoffer, Dave Modiano, Bob Haimes, Steve Allmaras and Richard Shapiro for stimulating technical discussions which resulted in a better development of my code and techniques of presenting the graphics output in an appropriate manner.

I would like to thank my wife, Van Pervaiz, for her relentless efforts in making me happy for all the years that we have been married. She had been extremely supportive during the final few months when I had to work nights on a regular basis. We are happy that this effort is finally over and we can now spend some time before she could undertake her graduate studies.

This work was supported by NASA under Grant NAG-1-507, and the work was supervised by the technical monitors Dr. James C. Townsend and Mr. N. Duane Melson at various stages.

Contents

List of Figures

List of Tables

Nomenclature

a_f	local frozen speed of sound, Chapter [4,6,7,A]
a_s, b_s	constants in specific heat model, Chapter [2]
A	cross-sectional stream-tube area for quasi 1-D flow, Chapter [2,8]
A_C	area of cell C, Chapter [3,6]
A_m	amplitude of oscillations, Chapter [8]
A_s	chemical symbol for species s, Chapter [2]
A_{cr}	pre-exponential factor for equilibrium rate constant in reaction r, Chapter [2]
A_{fr}, A_{br}	pre-exponential factor for forward, backward rate coefficient in reaction r, Chapter [2]
A_m, B_m, C_m	thermodynamic coefficients for evaluating temperature, Chapter [2]
C_{f_d}	maximum fraction of cells that can be embedded, Chapter [5]
C_s	concentration of species s, Chapter [2]
C_{ab}	correlation coefficient between components a and b of spatial adaptation criterion variable, Chapter [5]
C_{v_s}	constant volume specific heat for species s, Chapter [2,7,A]
C_{p_s}	constant pressure specific heat for species s, Chapter [2,7,A]
CF_r	choking factor for reaction r, Chapter [4]
D	grid Damköhler number, Chapter [4]
D	driving force for determining flux balance, Chapter [6]
$\mathcal{D}$	diagonal cell dimension, Chapter [4]
D_{ij}	binary diffusion coefficient for species i and j, Chapter [2]
D_s	diffusion coefficient of species s in the mixture, Chapter [2]
E	specific internal energy of gas mixture, Chapter [2]
E_{cr}	activation energy for equilibrium rate constant in reaction r, Chapter [2]
E_{fr}, E_{br}	activation energy for forward, backward rate coefficient in reaction r, Chapter [2]
f/a	fuel to air ratio by mass, Chapter [2]
F, G, F_i, G_i	flux dyadic vectors and their components, Chapter [2,3,4,6,7,A]
$\mathbf{F}_s$, f_{s_i}	external force per unit mass on species s and its components, Chapter [2]
F_U, G_U, W_U	Jacobian matrices, Chapter [3,7,A]
g_s	specific Gibbs free energy for species s, Chapter [2]

$\bar{g}_s$	partial molal Gibbs free energy or chemical potential for species s, Chapter [2]
G	amplification factor, Chapter [4]
G	total Gibbs free energy for the mixture, Chapter [2]
h	specific enthalpy of the mixture, Chapter [2]
h_s	specific enthalpy of species s, Chapter [2]
H_{f_s}	standard specific heat of formation for species s at temperature T_0, Chapter [2,A]
I	square root of -1, Chapter [4]
$\bar{\bar{I}}, I$	identity matrix or unit tensor, Chapter [2,6,A]
J	Jacobian determinant of transformation, Chapter [2,3]
$\mathbf{J}_s$	diffusion flux of species s in the mixture, Chapter [2]
k	thermal conductivity coefficient for the mixture, Chapter [2]
K_{fr}, K_{br}	forward, backward rate coefficient for reaction r, Chapter [2,4]
K_{cr}	equilibrium constant for concentrations for reaction r, Chapter [2]
K_{pr}	equilibrium constant for partial pressures for reaction r, Chapter [2]
$\mathbf{L}$, L	left eigenvector of matrix F_U, Chapter [7,A]
L_r	reference length scale, Chapter [2,8]
m	size of a time-stride or current maximum allowable temporal level of cells, Chapter [6]
$\hat{m}$	molecular mass of the mixture, Chapter [2,A]
$\hat{m}_s$	molecular mass of species s, Chapter [2,4,7,8,A]
M	diagonal matrix of mean values for the refinement parameter, Chapter [6]
M	prescribed maximum temporal level of cells, Chapter [6]
M_s, M_f	shock Mach number, Chapter [7,8]
n	current temporal level of cells in a certain integration pass P, Chapter [6]
$\hat{n}$	unit normal vector, Chapter [3,7]
n_s	number of moles of species s in the mixture, Chapter [2]
N_b	number of boundary points, Chapter [3]
N_c	number of cells, Chapter [3,5,7]
N_e	total number of equations to be solved, Chapter [5,7,A]
N_n	number of nodes, Chapter [3]
N_q	number of components in the spatial adaptation criterion variable, Chapter [5]
N_x	total number of layers of extension cells, Chapter [5]
p	hydrostatic pressure, Chapter [2,7,8,A]
p_0	a fixed reference pressure for chemistry, Chapter [2]
P_C	perimeter of cell C, Chapter [3]

p_s	partial pressure of species s, Chapter [2]
P	an integration pass, $P \in [1, P_T]$, Chapter [6]
P_T	total number of integration passes, Chapter [6]
q	parameter indicating inclusion or exclusion of second order source terms, Chapter [3,4]
$\mathbf{q}$, q_i	heat flux vector and its components, Chapter [2]
$\mathbf{q}_r$, q_{r_i}	radiation flux vector and its components, Chapter [2]
Q, q_i	spatial adaptation criterion vector and its components, Chapter [5,8]
Q, q_i	a characteristic variable, LU, and its components, Chapter [7]
r^2	refinement parameter for spatial adaptation, Chapter [5,8]
R	total number of reactions, Chapter [2,3,4]
R	"characteristic flux vector", LG, Chapter [7]
R_d, R_{d1}, R_{d2}	divide threshold limits for spatial adaptation, Chapter [5]
R_{min}, R_{max}	minimum and maximum refinement parameter values in the spatial domain, Chapter [5]
$\mathcal{R}$	universal gas constant, Chapter [2,7,8]
s	specific entropy, Chapter [2]
s	curvilinear coordinate, Chapter [3]
(s, n)	natural coordinate system, along and normal to a streamline Chapter [7]
s_{ab}	covariance between components a and b of a vector, Chapter [5]
S	total number of species, Chapter [2,3,4,A]
$S_s, \acute{S}_s$	Sutherland constants, Chapter [2]
t	time coordinate, Chapter [2,3,4,6,7,8]
T	temperature of the mixture, Chapter [2,4,7,8]
T_0	a fixed reference temperature, Chapter [2,A]
T_s^ϵ	effective temperature for molecular diffusion of species s, Chapter [2]
T_{sj}^ϵ	effective temperature for the computation of binary diffusion coefficient between species s and j, Chapter [2]
u, v, u_i	components of velocity vector, $\mathbf{V}$, in the x_i direction, $i = 1, 2, 3$, Chapter [2,4,6,7,8,A]
U, U_i	state (conservative, dependent variable) vector, and its components, Chapter [2,3,4,5,6,7,A]
u	dependent variable for a scalar model equation, Chapter [4]
V	volume, Chapter [3]
$\mathbf{V}, V$	mass average velocity vector and speed of the gas mixture, Chapter [2,7,A]
$\mathbf{V}_s$, V_{s_i}	diffusion velocity vector for species s and its components, Chapter [2]

w	fluid velocity in a frame of reference attached to a moving shock, Chapter [7]
W	source vector, Chapter [2,3,4,6,7]
$\dot{w}_s$	molal production rate of species s by all reactions (mole/vol/time), Chapter [2]
$\dot{W}_s$	mass production rate of species s by all reactions (mass/vol/time), Chapter [2]
x, y, x_i	spatial coordinates, $i = 1, 2, 3$, Chapter [2,3,4,6,7,8]
x_t	relaxation length, Chapter [7]
Y_s	mass fraction of species s, Chapter [2,4,5,6,7,8,A]
Z	chemical symbol for Lightill dissociated atom, Chapter [2,4]
Z, z_i	"characteristic source vector", LW, and its components, Chapter [7]

α	angle which the solid wall makes with x-axis, Chapter [7]
α_{sr}	stoichiometric coefficient for reactant species s in reaction r, Chapter [2,4]
α'_{sr}	order coefficient for forward reaction r and species s, Chapter [2]
β	bulk viscosity coefficient of the mixture, Chapter [2]
β_{sr}	stoichiometric coefficient for product species s in reaction r, Chapter [2,4]
β'_{sr}	order coefficient for backward reaction r and species s, Chapter [2]
γ, γ_f	ratio of specific heats, Chapter [2,6,7,8,A]
Γ	CFL number, Chapter [4,6,8]
Γ	the ratio $(\gamma - 1)/(\gamma + 1)$, Chapter [7]
δ	a constant in the artificial viscosity formulation, Chapter [3]
δ_{ij}	Kronecker delta, Chapter [2,7]
Δt	time-step, time-stride, Chapter [3,4,6,8]
ϵ	total internal energy per unit volume, Chapter [2,7,A]
ϵ_0, ϵ_1	small positive numbers used in evaluating temporal threshold of cell changes, Chapter [6]
ϵ_j	non-uniformity parameter for one spatial dimension at node j, Chapter [3]
η_{cr}	temperature exponent for equilibrium rate constant in reaction r, Chapter [2]
η_{fr}, η_{br}	temperature exponent for forward, backward rate coefficient in reaction r, Chapter [2,4,7]
θ	phase angle, Chapter [4]
θ_d	characteristic dissociation temperature for Lighthill model,

	Chapter [2,4,7]
κ_C	normalized scaling for artificial viscosity coefficient for cell C, Chapter [3]
λ	second coefficient of mixture viscosity, Chapter [2]
λ_j	j^{th} eigenvalue, Chapter [7,A]
λ_r	degree of reaction r, Chapter [2]
λ_{min}	a pre-selected non-dimensional negative number, Chapter [4]
Λ	diagonal matrix with entries equal to the eigenvalues of a flux vector, Chapter [7]
μ	(first) coefficient of shear viscosity or dynamic viscosity, Chapter [2]
μ_a	average of a component a for a vector over all nodes, Chapter [5,8]
ν	kinematic viscosity, Chapter [2]
ξ, η, τ	computational coordinates, Chapter [2,3]
ρ	mixture density, Chapter [2,3,4,5,6,7,8,A]
ρ_d	characteristic dissociation density for Lighthill model, Chapter [2,4,7,8]
σ_j	artificial viscosity coefficient at node j, Chapter [3]
σ_s	effective diameter for the molecule of species s (Angstrom units), Chapter [2]
σ_{sj}	effective collision diameter between molecules of species s and j (Angstrom units), Chapter [2]
$\sigma_{min}, \sigma_{max}$	minimum and maximum artificial viscosity coefficient in whole domain, Chapter [3]
Σ, s_{ij}	variance covariance matrix and its components, Chapter [5,8]
τ	chemical time-scale, Chapter [4]
$\overline{\overline{\tau}}, \tau_{ij}$	stress tensor and its components, Chapter [2]
ϕ	equivalence ratio, Chapter [2,8]
ϕ	a scalar variable, Chapter [3,5,7]
ϕ_r	phase shift, Chapter [4]
$\phi_{sj}, \acute{\phi}_{sj}$	Wilke's binary dimensionless ratios between species s and j for the computation of viscosity and thermal conductivity of mixture, Chapter [2]
Φ	reaction parameter for Lighthill model, Chapter [2,4,7,8]
Ψ_{jC}	artificial viscosity contribution at node j due to cell C, Chapter [3,4]
ω	wave number, Chapter [4]
ω	frequency of oscillations, Chapter [8]
$\omega_s(T)$	a term in specific Gibbs function for species s, Chapter [2]
$\Omega, \partial\Omega$	region and boundary of integration, Chapter [3]
Ω_D	dimensionless collision integral, Chapter [2]

Ω_{fr}, Ω_{br}	forward, backward progress rate of reaction r, Chapter [2,4]

Subscripts

∞	free stream conditions
0	reference chemistry value or value at time 0
e	local equilibrium value
e	exit conditions
f	local frozen value
i	a component of a vector (U, F, G, W) $i = 1, \ldots, N_e$
i	inlet conditions
i, j, k, l	node locations
r	chemical reaction index and reference condition
s	chemical species index
t	tangential component
U	Jacobian with respect to state vector
x, y	along streamwise or transverse direction
ξ, η	along ξ or η direction

Superscripts

$*$	non-dimensional variable
$\tilde{}$	transformed variable
c	corrected value in predictor-corrector approach
n	time level
p	predicted value in predictor-corrector approach

Chapter 1

Introduction

The field of Computational Fluid Dynamics (CFD) has evolved during the past two decades to an extent that computational models are playing an increasingly important role in the design of aerodynamic vehicles. This rapid evolution of CFD is prompted by increasing costs associated with experimental design and decreasing costs of computer hardware, as well as the detailed behavior that can be determined when the relevant physics can be modelled. The design of hypersonic vehicles and their engines, for example, demands some sort of modelling to account for real gas effects. However, calculations involving non-equilibrium reacting flows can be an order of magnitude more expensive than corresponding frozen flow solutions. The computational cost increases as more realistic multi-component and multi-reaction systems are considered. The costs increase even further if effects of vibrational and electronic non-equilibrium, radiation, plasma dynamics, non-ideal equations of state, condensation and ablation, realistic models for diffusion coefficients for multi-component systems, *etc.*, are considered.

This thesis is concerned with fluid dynamics involving the simultaneous occurrence of chemical reactions and convection of mass, momentum and energy for both steady and transient situations. Chemical kinetics pertaining to finite rate chemistry introduces non-equilibrium features which interact with the classical fluid rate processes. For the numerical examples described here the effects of viscosity, diffusion and heat transfer (transport effects) are neglected and the flow description is based upon Euler equations and species conservation equations in conservative form. Quasi 1-D and 2-D flows are considered with multiple number of reactions. A number of examples in one spatial dimension are used as vehicles to demonstrate certain important concepts and to illustrate specific analytical and numerical techniques.

1.1 Motivation

The importance of understanding the interactions pertaining to chemically reacting flows has recently become of paramount nature due to the renewed interest in hypersonic flows and advanced aerospace propulsion systems. A concerted effort is now directed towards the research and development of the National Aero-Space Plane or NASP. The hydrogen fueled scramjet (supersonic combustion ramjet) is regarded as a strong candidate for propelling such a hypersonic transatmospheric vehicle. The design of such an engine demands understanding the fluid dynamics of hydrogen-air combustion and flame-holders over a range of flow conditions. The high temperature non-equilibrium effects of chemical reactions associated with the re-entry of Orbital Space Shuttle or similar hypersonic vehicles is not fully understood. Accurate numerical modelling for these situations can provide valuable insight into the nature of reacting flows. Other areas of related real gas interest are rocket plumes, aircraft signatures, materials ablation under lasing action, gaseous radiation effects, *etc.* The aerodynamic processes governing such reacting flows are exceedingly complex and can involve strong interactions between chemical and fluid dynamical effects.

Chemically reacting flows often require lengthy computations due to a larger number of descriptive conservation equations which correspond to multi-component species in multiple non-equilibrium reactions. The calculations are particularly costly due to the stiffness introduced by finite rate chemical kinetics with appreciably different time-scales. These factors are the motivation for a search for more efficient and accurate algorithms for reacting flows. For example, the concept of ***equation adaptation, i.e.,*** introducing a simpler set of equations under special conditions, can be used when there are sub-domains of frozen flow in an otherwise relaxing flow system. The expensive calculation of source terms and their Jacobians may be avoided when the static temperature remains below a pre-specified threshold temperature, since the contribution from such terms is negligible compared to the convective terms. For these frozen flow situations the chemical time-scales become large compared to the convection time-scale. By-passing the chemical source term manipulations is almost as effective as solving a

system with only four conservation equations, because the major costs associated with the computations of reacting flows are the calculations involving the chemical source terms.

The accuracy of computer simulations depends in part upon the size of computational cells in space and time and also on the accuracy and stability of the numerical algorithm. The limitations pertaining to computer resources for adequate spatial and temporal resolution has led to the desire for performing adaptation in both space and time. The resolution limiting restrictions are primarily imposed by cost considerations and computer hardware constraints such as insufficient computer memory, insufficient data storage facilities and slow processing speed [100]. The term ***spatial adaptation*** is associated with the description of the numerical procedures that automatically assign finer spatial cells in the regions of interest [15,33,55,130]. The regions of added resolution delineate ***features*** which are detected by examining those cells characterized by steep local changes [35] or high truncation errors [13]. Spatial adaptation generally results in smaller cell dimensions in regions where these features cluster and coarser cells in relatively uniform flow regions.

The concept of spatial adaptation can be extended to temporal adaptation, with an allowance for spatial variation of the cell time-steps so as to avoid the severe and costly constraint of a globally minimum time-step. For the procedure presented here it automatically increases the temporal resolution in the regions of large temporal gradients of some pre-selected variables. Thus the concept is similar to its spatial counterpart in the sense that smaller time-steps are taken in the regions of local rapid adjustments. The utilization of variable time-steps for unsteady flows through temporal adaptation will be demonstrated to be an efficient way of handling time-differencing. The temporal procedure results in multiple integration passes for cells with smaller time-steps but eventually all the cells arrive at the same time-value. The time difference between two consecutive isotemporal surfaces is the maximum time-step allocated to any cell and is referred to here as *time-stride*.

1.2 Past Studies

It is relevant to take note of earlier work related to supersonic finite rate processes from an analytic, design and computational point of view. The present work's concern with adaptation procedures for unsteady problems relates also to efforts dealing with unsteady blast waves, stiffness and mesh enrichment. Typical sources for the basic governing equations are Degroot and Manzur [40], Toong [131] and Williams [141].

The phenomenon of supersonic combustion has been observed and known for more than a century. The concept of detonative combustion originated in the latter part of the nineteenth century when French chemist Le Châtelier noted that some combustible mixtures under certain conditions developed combustion waves which possessed extraordinarily high velocities. About 1900 Chapman [28] and Jouguet [70] independently proposed explanations for such phenomenon. They suggested that detonations can be regarded as shock waves followed by combustion which is triggered by the high temperature aft of the shock rather than the diffusion processes usually associated with deflagrations [92].

Much more recently a number of studies have been applied to scramjets. These scramjets are advocated to provide a viable propulsion option for flight speeds in excess of Mach 5 [6,69,139]. For such vehicles the combustor is integrated into the airframe; the vehicle itself provides the engine with hot compressed air through inlet shocks and expansion through a streamlined exhaust while keeping the drag associated with the engine to a minimal. Analysis predicts that the contribution of the vehicle forebody and afterbody can be responsible for up to 70% of the net thrust [62].

The overall work has involved examination of engine design concepts, simple analytic techniques for evaluating performance, experimental investigation to provide critical design information and numerical solutions to provide detailed insight into the implied reacting flows. Dugger *et. al.* [45] have examined the performance of a ramjet engine by employing a constant pressure supersonic heat addition behind flame-induced oblique shocks. Morrison [92,93] analyzed the *oblique detonation wave ramjet's* performance

for varying stoichiometric hydrogen-air equivalence ratios and a range of flight Mach numbers from 6 to 16. In the oblique detonation wave ramjet the compression process is moderate and carried out to relatively low pressures and temperatures; the detonative process supplies additional compression and high temperatures for combustion in very short length scales. In addition to ramjets characterized by standing detonation waves, there are *diffusive burning scramjets* in which the compression process in the inlet is carried out to high pressure and temperature for reaction to occur in relatively larger spatial domains. For these, the compression or diffusion process is commonly treated separately from the combustion process for the purpose of analysis. Billig [16] provided guidelines for the design of various inlet geometries for scramjets. Northam and Anderson [98] discuss the design philosophy of the NASA Langley's fixed geometry airframe-integrated modular scramjet; an extensive bibliography is provided ibidem. Other studies that discuss the analytic and design aspects of the scramjet concept are [1,21,83,120]. Since a design procedure generally involves repetitive computations, which vary the parameters influencing design or evaluate new design concepts, an efficient algorithm for such calculations would be very beneficial.

Past numerical studies on supersonic reacting flows have been quite limited. The main reason had been the limitations in computer resources in providing a description for reasonably detailed models. Although a number of strides have been made in computer architectures in the recent past, it is still not possible, for example, to provide a numerical solution of a complete engine which takes into account reactions, turbulence, unsteadiness, *etc.* Therefore it is still desirable to study flow fields on a component basis and at the same time utilize efficient and inexpensive algorithms. Drummond [41] has examined transverse fuel injection through a 2-D slot in a scramjet engine using mass diffusion terms but in the absence of chemical reactions, and utilized an algebraic turbulence model. The results show a small separated region in the vicinity of the injector and he speculated that ignition would commence in this region. Drummond and Weidner [44] have considered the mixing of transverse and parallel streams of air and fuel in a converging-diverging channel with embedded struts which eject the fuel. A complete reaction model was used for the hydrogen fuel. The calculations again indicate a small separated region near the injectors where significant reaction occurs. The pri-

mary reasons for this are the subsonic conditions and the complete nature of the reaction model. Griffin *et. al.* [59] have considered injection of parallel fuel-rich exhaust in an axisymmetric geometry while utilizing a Parabolized Navier-Stokes (PNS) code with a local, diffusion controlled, chemical equilibrium system. This paper also discusses some inlet design aspects and ramjet combustion modelling. The radiation effect for gray and non-gray models are studied for simple geometries in References [84,85], whereas direct simulation Monte Carlo method coupled with a dissociating and ionizing gas model with thermal radiation is considered in Reference [94]. Although the current research does not address these issues, such references are cited here to indicate the diversity and complexity of the hypersonic flows and to emphasize that if quantitatively accurate simulations are desired, then all the pertinent physics must be taken into account. However, currently a comprehensive numerical analysis is not possible and hence research efforts should be directed in designing more effective modules for specific physical aspects which could eventually be integrated. Other references that have employed numerical simulations in hypersonic reacting flows are [12,18,25,37,38,46,48,56,67,73,89,117,142,146]. The bibliography provided here is by no means complete, it represents only a small fraction of the studies that have been carried out. A detailed list can be found in the survey papers of References [11,98,139].

Knowledge is limited as well with respect to the dynamics of unsteady (whether reacting or frozen) flows. It is important to understand flow fields in response to temporally varying conditions. There are relevant questions about such inflow conditions and their influence on the rest of the flow field. Other questions pertain to the influence of an oscillating fuel supply on flame stability. Kumar *et. al.* [75] have considered one such case and examined an oscillating shock interaction with a scramjet combustor utilizing a simplified combustion model. Another area where unsteady flows are involved and need further study, pertains to the propagation of detonations in gases and their interactions with stationary objects. Among many studies, blast wave interactions have been considered by [3,126,127,135,137,144,145,147].

The phenomenon of numerical stiffness pertaining to chemical source terms has been known since the early fifties. One of the first algorithms to cope with the difficulties of

integrating stiff ordinary differential equations was suggested by Curtiss and Hirschfelder [31] for chemical kinetics studies. Dahlquist [32] indicated numerical instability as the cause of the difficulty and provided basic definitions and concepts that are useful in classifying and evaluating algorithms from a stiffness perspective. A detailed account of stiffness can be found in the text by Gear [54] and a number of survey papers have recently appeared — typical examples being Bui *et. al.* [22], Enright and Hull [50] and May and Noye [88]. Radhakrishnan [109] has compared a number of stiff and nonstiff methods. Applications of the approach to systems of partial differential equations have been carried out by Bussing [23,24], Drummond [42], Rivard [114] and Stalnaker *et. al.* [122].

Recently a number of studies have been carried out on mesh enrichment to capture local features via spatial adaptation and thus concentrate computing resources where they are needed most. The techniques have been applied to elliptic [130], parabolic [97] and hyperbolic [13] systems of equations with typical references as indicated in these areas. There are studies in which the adaptive grid nodes are placed according to variational, finite-element formulation [55,80,95,104]. Methods in which the overall computational domain is subdivided into independent zones with non-overlapping or patched grids have received attention in [63,110]. Other methods redistribute and/or cluster grids in the vicinity of known features [7,19,47,49,58,66]. An alternate approach is to do successive local embedding without moving the grids [15,35,99,128]. In this approach rectangular fine grids are superimposed on an underlying coarse grid in those regions where solution accuracy is inadequate. Berger [13,14] bases the refinement decision on the estimates of local truncation errors by utilizing Richardson extrapolation. Dannenhoffer and Baron [36,34,35] base the refinement on first differences of density for transonic applications. This locally embedding approach is the basis of spatial adaptation in this thesis and is discussed in detail in Chapter 5. Very few adaptive procedures have been applied to reacting flows, References [106,121,125] have considered adaptation in one spatial dimension and Reference [105] has applied the embedded mesh approach to two spatial dimensions. An extensive list of papers concerned with spatially adaptive grids can be found in the survey papers of [9,129].

In addition to refinement in space, grids may be refined in time as well, so that smaller time-steps are taken on spatially fine grids or where rapid changes occur. For frozen flow applications this is generally done by keeping the CFL (Courant-Friedrichs-Lewy) number nearly the same on coarse and fine grids [14], so that the same integrator is stable on each grid. The smallest time-step does not have to applied on the entire grid. Although a number of adaptive examples have been carried out for unsteady flows [80,81,104,105,111,143], most of these applications have been performed by utilizing global minimum time-steps. Osher and Sanders [101] have discussed a conservative temporal interface formulation that links together an arbitrary number of space regions containing fine and coarse time increments in one spatial dimension. The interface difference equations are formulated in a predictor-corrector form and it seems that their generalization to include additional topologies for two spatial dimensions would be complicated. They have also proved that utilizing a variable step time-differencing leads to correct physical solution for a scalar, monotone discretization in one spatial dimension. Löhner *et. al.* [81,82] have proposed a domain splitting technique to advance the solution with different time-steps on different portions of the mesh for multi-dimensional problems. These references also propose an integration sequence for cells in regions of time-steps that differ from global minimum values by integral multiples. The temporal interfaces are handled by regarding two layers of cells to be a part of both temporally fine and coarse regions and applying interface conditions at the boundary nodes of these layers. The interface conditions depend upon advancement of the time-steps in regions of temporally fine or coarse resolution. A similar integration sequence is proposed in this thesis for cells characterized by different temporal levels and the interface conditions are applied in the spirit of cell by cell integration and as such temporal interfaces enter into the calculations only at the time of updating of the state vectors. The details of temporal adaptation and interface manipulations are presented in Chapter 6.

1.3 Present Work

The objectives of the present study are three-fold. Firstly, to examine predominantly supersonic reacting flows in which the transport effects may be neglected. Secondly, to perform spatial adaptation in regions of large spatial non-uniformities. This aspect is applicable to both steady and unsteady flow situations. The third objective is to perform temporal adaptation, for certain unsteady applications. Emphasis is placed on understanding supersonic combustion of hydrogen in air and moving blast waves in dissociating gases. The unique part of the work, relative to previous studies, is the coupling of spatial and temporal adaptation procedures for chemically reacting systems. A computer program entitled STAR (Spatio-Temporal Adaptive Reactive) Code has been developed as a part of this effort that implements the concepts that have been developed. The procedure is referred to as the spatio-temporal adaptive algorithm.

The following sub-sections provide some justification for using an Euler system of equations for the problems considered here and why spatial and/or temporal adaptation is important. This is followed by an overview of the spatio-temporal adaptive procedure as applied in this thesis.

1.3.1 Why Use Euler Equations?

The immediate result of the cost factor appears as constraints on the software. It is less expensive to carry out potential flow calculations compared to Euler equations which in turn are relatively cheaper than a system involving the transport effects of viscosity, heat transfer and species diffusion. The costs are not associated just with complex models, but that the resolution requirements for both space and time increase with the modelling of additional physics. For example it will be a waste of effort to solve for Navier Stokes equations on the same sort of grid as one would typically use for inviscid flows. For most examples presented in this thesis, using potential flow solver would be inappropriate since the rotationality associated with strong shock structures for supersonic and hypersonic flows would not be captured correctly. However, since the

transport effects are usually limited to regions whose typical dimension (*e.g.*, boundary layer thickness) is generally small compared to the reference dimension (*e.g.*, chord length), the Euler equations can be easily used to understand salient features of these flows.

One of the concerns in combustion applications is that streams of reactants may be impinging or flowing parallel to one another and at the same time mix under the action of differing momenta and molecular diffusion and hence reacting to form products. Thin viscous shear layers are important in determining the location of separation and the generation of vorticity in the flow. However, for the predominantly supersonic streams in this thesis, the diffusion effects are still limited to small regions in the vicinity of slip surfaces. Furthermore, as has been experimentally observed by Papamoschou and Roshko [102] for supersonic mixing layers, the shear layer spreading is about one quarter that of an incompressible layer at the same ratios of velocity and density. Hence Euler equations can be used, without serious misgivings, for these flow situations. Although the capability of solving the full 2-D Navier Stokes equations, on a cell by cell basis, has been added to the STAR code, this capability is not tested on a wide variety of problems and extension to include turbulence modelling has not been done.

1.3.2 Why Use Adaptive Grids?

It is well-known that greater accuracy is realized when finer grids are utilized in both space and time. This is because the truncation error of the numerical schemes is dependent upon fineness of the cells; with increasingly finer cells this error tends towards zero. It is also well-established that an accurate description of small structures in a flow can be realized generally by spanning the structure with an appropriate number of computational cells. The uncertainty pertaining to the location of a particular feature within a cell of course could be reduced by increasing spatial resolution. If the flow structures are not adequately resolved, they become numerically diffused since a discrete model inherently spreads flow discontinuities over several cells and thereby degrades accuracy. Hence spatial resolution is essential near features like shocks, relaxation zones,

vortices, slip lines, *etc.*

The classical way to provide adequate resolution for the capture of features is to use globally fine grids. This usually results in a tremendous number of cells which places extensive demands on the CPU memory. Furthermore, global refinement can result in prohibitively time-consuming computations and hence is not a very attractive option. The loss of efficiency can be countered by the use of adaptive gridding techniques. The spatial adaptation approach utilized in this thesis locally divides the cells to yield additional resolution near features characterized by large spatial non-uniformities. This approach is discussed in detail in Chapter 5 and follows the procedure presented by Dannenhoffer [33]. The extensions include utilization of multiple variables in deciding on regions of added resolution and a procedure for adding multiple layers of buffer zones to spatially embedded regions. The adaptive embedding algorithms have the advantage that meshes are refined only where necessary and as the solution evolves, thereby providing accurate and relatively inexpensive solutions. Since the local embedding can be carried out in a recursive manner, very fine grid spacing can be maintained in the vicinity of the physical structures being captured. Furthermore, since the resolution is enhanced only locally at the features, with coarser grids near successively uniform flow regions, the computations with such grids consume significantly less computer resources than global refinement. There are substantial savings in both CPU time and memory.

It is clear from the CFL constraint that the resolution requirements in space generally imply a corresponding imposition on resolution in time. For most frozen flows this is the primary constraint. However, for reacting flows other temporal resolution requirements may be even more stringent than those implied by the spatial resolution. Similarly for moving blast waves the maximum eigenvalues across a shock can be an order of magnitude different. Hence the resolution in time may be controlled only in part by the resolution in space. For cases where strong coupling does exist between the two, allocation of temporal resolution simply follows from that of spatial resolution. For those cases, in two spatial dimensions, increasing the spatial resolution by a factor of four imposes a corresponding factor of two in time-steps; hence there is an eight-fold increase in computational work to advance to a given interval of time.

In chemically reacting flows, the computations of chemical kinetic terms is often more expensive than evaluations of convective and/or diffusive transport terms. The cost increases with the number of species, the number of reactions connecting these species, the number of spatial cells and the inverse of the time-step size. For flame and detonation simulations the overall calculation may take two or more orders of magnitude longer compared to frozen flow situations [100]. Calculations may also be costly due to stiffness introduced into the equations by including finite rate chemical kinetics which are necessary to describe the physical situation.

When the reactive equations are stiff in the sense that numerical stability rather than accuracy dictates the time-steps, then an implicit scheme can be used to partially alleviate the computational overheads. The implicit approach presented here utilizes the concept of Newton-Raphson expansion of the source terms as proposed by Bussing [23] for steady state applications. However, for unsteady flows the time-steps must be appropriately small to resolve the features involving local rapid chemical adjustments. These are generally changing patterns of resolution requirements as the rapid transients form, gather strength, interact and deform other flow features and eventually decay in different periods and positions. Hence there are conflicting requirements on unsteady reacting flows in the sense that for efficiency the advancing time-steps may have to be reduced in certain portions of the space-time domain where adjustments occur and a utilization of longer time-steps be made where there are negligible temporal gradients.

Just as different spatial resolutions are allocated at different locations within a spatial grid in order to achieve CPU time gains, it would be beneficial to take advantage of the large spatial variations of time-steps for reacting flows. In fact gains due to utilization of different time-steps also can be achieved for unsteady frozen flows if there exist substantial variations in spatial cell volumes, which indeed may well be a result of spatial adaptation. Similarly for moving blast waves the eigenvalues involved in the CFL constraint may change substantially across the shock that may result in a corresponding variation of time-steps across this shock even for spatially uniform grids. An efficient time-differencing technique is developed in this thesis that makes possible advancement of cells on a step-size which is a multiple of a global minimum time-step. Without this

technique the severe and costly constraint associated with a globally minimum time-step would be applicable for time accuracy and computational costs would be literally immense. In this technique the cells with the same time-step are integrated and updated together on different integration passes of the temporal adaptation cycle but the majority of small time-step cells fall in only a small portion of the overall space/time domain. Once all integration passes are completed for each time-stride unit, all nodes in the domain arrive at the same time-station.

1.4 Overview of Adaptive Procedure

Before the application of numerical solution to a problem, it must be decided whether interest is restricted to a steady state limit or that an unsteady approach is relevant. Steady state problems may involve local time-stepping, multiple grids and other acceleration techniques, whereas for unsteady flows such techniques are clearly inapplicable. It is suggested that temporal adaptation would be more appropriate for the unsteady case. Spatial adaptation is beneficial for both approaches; however, for unsteady flows, spatial adaptation procedure must be applied frequently because the features to be resolved may be moving and the adaptive grid clearly must track these features at a synchronous speed. For such unsteady flows the spatial adaptation procedure may have to be applied after the completion of each and every time-stride. For the steady state the stationary features require an adaptive procedure only occasionally and the number of such operations generally equals the number of the spatially embedded levels desired for the cells. In such cases the adaptive procedure is generally applied after the residuals have subsided below a pre-determined level.

It is not imperative to do reverse embedding for steady state applications; however, it does become necessary for unsteady flows to allow for a cell fusion capability since otherwise grids may become uniformly fine after a while and the advantage of dynamic embedding would be lost. Since the rate of change of flow features may be very large for certain unsteady applications, it is necessary to extend the spatially resolved region by a certain number of cells to ensure that the flow features will remain within this

resolved region during the next time-stride unit. There is no such need for steady state flows due to the stationary nature of the flow features.

For steady state applications there clearly is no need to have adequate time-step resolution and implicit schemes involving large time-steps which alter the transient history may be used. This is obviously inappropriate for unsteady situations, although implicit schemes which only limit the time-steps in the regions where dynamic changes occur may be used.

The choice of initial grid conditions is especially important for an unsteady flow. If large spatial gradients are present in the initial flow field and the spatial grid is coarse in their vicinity, the initial integrated solution will be degraded and will propagate as such to other spatial locations at later time levels. Of course, this is not as important in cases which lead to dynamic unsteady periodicity for large times. The subdivision of meshes necessitates assignment of state vector at the newly created nodes. A polynomial interpolation of these initial values based upon the surrounding nodes may be inconsistent with the initial condition. For example a shock tube problem suggests that finer cells be inserted near the contact discontinuity surface; a linear interpolation for nodes bordering this initial step function would degrade the step function. The procedure which involves care in assigning the initial values at the newly created nodes is referred to as *pre-embedding* which is frequently performed prior to the execution of the integration process. Pre-embedding is unimportant for steady state flows and any interpolated values may be used at the newly created nodes.

The spatio-temporal adaptive algorithm discussed in this thesis is summarized in this paragraph and it generally assumes that unsteady flow problems are under consideration. The algorithm periodically examines the evolving numerical solution, applies spatial adaptation to the existing grid, determines an appropriate time-stepping sequence for each cell in order to make up consistent time-stride units for the entire domain, and finally integrates the equations. The spatial adaptation involves the detection of regions of large spatial non-uniformities and subsequent subdivision of the corresponding grids. Reverse embedding to a coarser mesh is allowed up to the initial

coarsest level global grid. When the initial flow field on a coarse grid involves spatial non-uniformities, consistent pre-embedding is applied without degrading this initial field. In a similar manner the temporal gradients are monitored so as to maintain sufficiently small time-steps for adequate local resolution and stability. The time-step resolution takes into account the classical domain of dependence restriction and the requirement imposed by large non-equilibrium source terms. The spatial and temporal resolution requirements are generally coupled through the CFL restriction for frozen flows. This coupling also exists for the criterion which takes into account the variations of the source terms. The algorithm will now be described in somewhat more detail.

The procedure starts with the selection of a suitable global stationary grid in space and a provision of initial conditions on this grid. Pre-embedding may be needed for initial coarse grid in the regions involving large spatial non-uniformities. As noted earlier, pre-embedding is the same as spatial embedding except that the assignment at the newly created nodes is based upon the actual physical conditions at the initial time rather than the interpolated values from the nearby nodes (See Chapter 7). Spatial adaptation differs from pre-embedding in the sense that it is followed by subsequent integration of equations and may involve fusion of cells, whereas the objective of pre-embedding is to merely add enough resolution to the initial grid so that the gradients are appropriately represented without being diffused. The process of pre-embedding is generally repeated a number of times, until the desired spatial level of cells is achieved, before the execution of normal adaptive procedure can proceed.

Once the integration procedure is started, the evolving solution is examined for regions of relatively large gradients of some pre-selected criteria variables and the regions where these gradients exceed a threshold level, the grids are locally divided. Quadrilateral cells in two spatial dimensions are used for this purpose and the refinement of a cell is accomplished by dividing the cell into four subcells. Alternatively, when associated gradients diminish on a previously refined grid, and become less than another critical limit, those contiguous grids may be collapsed while making certain that the cells to be merged are those from the same parent cell. The initial (coarse) global grid is kept fixed by insisting that the coarsest cells (spatial level zero) be never merged to a coarser

state, no matter how smooth the evolving solution proves to be. In summary, the spatial adaptation procedure comprises of the following sequential operations (1) local embedding or cell division, (2) extension of spatially embedded regions, (3) coarsening or cell fusion in other regions, and (4) removal of the knottiness in the grid by avoiding islands and voids.

After the alterations are completed in the spatial grid structures, a sequence of time-steps is determined for all the cells in the domain. The cells with the same time-step are integrated and updated together on different integration passes of the temporal adaptation cycle. Once all the integration passes are completed, all the nodes in the domain arrive at the same time value (time-station) and a time-stride is completed. Depending upon the rate of variations of the flow features, the spatial adaptation may follow after this temporal adjustment or a number of time-strides may be carried out prior to the next spatial adjustment of the grids. The number of time-strides between two consecutive spatial adaptation procedures is user-controlled and is not dynamically computed by the algorithm, since this is a complicated business and is highly problem dependent. The user is generally aware of an expected rate of variations of feature properties and s/he could simply ask for the spatial and temporal procedures to alternate each other in a worst scenario. The integration of the equations continue until a desired number of time-strides is completed or when the time-level exceeds some user-supplied value.

1.5 Overview of the Thesis

This thesis describes the explicit and implicit numerical procedures and emphasizes the development of spatio-temporal adaptive techniques.

The conservative differential equations that govern the dynamics of reacting flow are outlined in Chapter 2. The equations are presented with the effects of viscosity, heat transfer and species diffusion included. The constitutive relations for the mixture properties are based on ideal mixture assumption. The mass action rate equations are

described by generalized Arhenius kinetics. The inviscid equations are then specialized for Cartesian and generalized coordinate systems and the normalization is discussed. The determination of temperature from the state vector is explained for a linear temperature model for constant pressure specific heats of individual species. The chapter concludes with a description of chemistry reaction models used in the thesis.

The finite difference equations and the solution method of the undivided grids is based on Ni scheme [96] and is described in Chapter 3. The difference equations for both one and two spatial dimensions are derived and the artificial diffusion model is explained. The treatment at the spatial interfaces, or the locations where the grid changes abruptly, is explained for two different approaches.

The difficulties encountered in the numerical solution of stiff chemical systems are presented in Chapter 4. The stiffness is examined for a linearized scalar source model in one spatial dimension and stability analysis is carried out. Two possible remedies to treat stiffness are presented; this may be accomplished by using first order implicit schemes or by using explicit schemes with the source terms modified in a particular manner.

Chapter 5 begins with an explanation of reasons why spatial adaptation is desired for computational models. The utilized data structure is detailed which allows rapid and efficient implementation of the spatial adaptation procedure. The methodology for the detection of flow features is based upon first differences of multiple components of spatial criterion vector. The scalar refinement parameter is based upon unbiased variabilities of these components and removes the correlation between individual components. The data structure details for grid division, grid fusion and the extension of spatially resolved regions is presented. The chapter concludes with the discussion on the avoidance of grid knottiness like islands and voids.

The concept of utilizing variable time-steps for solving time-accurate transient problems is developed in Chapter 6. It begins by examining the factors which limit the computational costs and the ways in which these costs can be reduced. The issue of temporal resolution is discussed in Section (6.2) for frozen and reacting situations. Illustrative

examples are given in Sections (6.3) and (6.4) for one spatial dimension and time-stride comprising of two time-steps. The temporal adaptation concept is generalized to include larger time-stride units in the last section.

The initial and boundary conditions are discussed in Chapter 7. The implications of both physical and numerical boundary conditions are described. The initial conditions include those for a shock tube and a moving shock and for a frozen or dissociating gas. An approximate characteristic analysis is presented for relaxing flows and is applied to subsonic inflow/outflow boundary conditions.

Chapter 8 contains the computational results. Selected examples for one and two spatial dimensions are presented for a perfect gas, Lighthill dissociating gas and Rogers and Chinitz [115] combustion model for hydrogen and air. The flow types include shock tubes, moving shocks, steady state and oscillating inflow.

Major conclusions are presented in the final Chapter 9. A discussion of possible extensions to the developed spatio-temporal adaptive algorithm is also presented.

Four appendices complete the thesis. Appendix A describes the details of the evaluations of Jacobians, eigenvalues and eigenvectors for the flux vectors. Important considerations which were taken into account while developing software are presented in Appendix B. A description of the utilized data structure from a coding perspective is given in Appendix C, this also includes a detailed description of the logic for cell division, fusion and buffer zone addition. The last appendix appears as a separate volume, that includes sample input files for the code, synopsis of computer variables in the common blocks and a listing of the code itself. Also included are graphics interface routines for generating plots based upon the data structure of the pointer system that was developed.

Chapter 2

Governing Equations

In this chapter the conservation equations for a general three-dimensional flow of a chemically reacting gaseous mixture are outlined. An ideal gas mixture is assumed, *i.e.*, the components of the mixture are regarded as perfect gases and Dalton's law holds for the mixture. Although effects pertaining to molecular transport phenomena have been neglected in this research, the terms describing such effects are retained in Section (2.2) for the sake of completeness and possible future extensions. The governing equations are presented in both vector and indicial tensor forms. The tensor form is useful in laying out the basic integration scheme whereas the vector form is important when using generalized curvilinear coordinates. For a detailed derivation of the conservation equations References [40,131,136,141] may be consulted.

The chapter starts with some introductory remarks and a description of full conservation equations in a 3D system. Section (2.3) summarizes the Euler equations for quasi-one-dimensional flow, whereas Section (2.4) summarizes the corresponding two-dimensional equations and discusses normalization. Section (2.5) discusses the Euler equations in a generalized transformed coordinate system. Section (2.6) explains the determination of temperature from the caloric equation of state. Section (2.7) discusses a general procedure of determining the equilibrium constants and suggests generalized Arrhenius form as a simple model. Finally Section (2.8) discusses the chemistry models used in the current research.

2.1 Introductory Remarks

Fluid motion is governed by the conservation of mass, momentum, energy, and species, various state and constitutive equations, and proper initial and boundary conditions. For a large class of situations, *irreversible flows* are described by linear functions of *thermodynamic forces*, as expressed by the so-called *phenomenological laws*. For example, Fourier's law of heat conduction expresses the heat flux as a linear function of temperature gradient. Similarly, Fick's law establishes a linear relation between the diffusion of mass and the concentration gradient. In a similar manner the phenomenon of thermal diffusion or *Soret effect* describes the diffusion of mass caused by temperature gradient. A reciprocal phenomenon, *viz.* the flow of heat resulting from concentration gradients is referred to as *Dufour effect* [40]. The effects pertaining to heat conduction, diffusion, *etc.* are often classified as *direct* whereas Dufour effect or thermal diffusion is labeled as *cross phenomenon* [131].

It is often projected by the CFD community that the advent of more powerful computers will allow routine solutions of the Navier-Stokes equations and therefore the need for doing experimental research will diminish. Navier Stokes equations are the subset of the actual fluid mechanic description that involve only direct linear modeling of some irreversible phenomenon and would not delineate situations dominated by other real effects. Although an attempt is made here to put forward equations describing the physics and chemistry of fluids with domain of application somewhat wider compared to the usual Navier Stokes equations, these equations are still limited in applications. Situations where these equations may be dubious will be pointed out as the need arises. The equations describing simulations have no bearing on nature itself, the limitations of the computational models are irrelevant so far as the experimental research is concerned. On the other hand there are restrictions on the experiments which may be non-existent while performing computations. It is the contention of the current author that computations will never replace experiments in their entirety; however, as our computational models will become more realistic these two approaches will be used in complementary rather than adversary roles.

2.2 Full Conservation Equations

For the equations of fluid motion the chemical mixture is assumed to be comprised of S species involved in R chemical reactions of the form

$$\sum_{s=1}^{S} \alpha_{sr} A_s \quad \rightleftharpoons \quad \sum_{s=1}^{S} \beta_{sr} A_s \qquad r = 1, \cdots, R \tag{2.1}$$

where α_{sr} and β_{sr} are the dimensionless stoichiometric coefficients for the s^{th} species in the r^{th} reaction, and A_s is the s^{th} participating molecule.

2.2.1 Continuity Equation

The global continuity equation in conservation form is

$$\frac{\partial \rho}{\partial t} + \sum_{j=1}^{3} \frac{\partial}{\partial x_j} (\rho u_j) \; = \; 0 \tag{2.2}$$

and the corresponding generalized vector form is

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) \; = \; 0 \tag{2.3}$$

where $\mathbf{V} = (u_1, u_2, u_3)$ denotes the velocity vector and ρ is the global (mixture) density.

2.2.2 Momentum Equations

The momentum equations in conservative tensor form are

$$\frac{\partial}{\partial t} (\rho u_i) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j} (\rho u_i u_j - \tau_{ij}) + \frac{\partial p}{\partial x_i} \; = \; \sum_{s=1}^{S} \rho Y_s f_{s_i} \qquad i = 1, 2, 3 \tag{2.4}$$

where the stress tensor components, τ_{ij}, for a Newtonian mixture is given by the following linear phenomenological law

$$\tau_{ij} \; = \; \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \delta_{ij} \lambda \sum_{k=1}^{3} \frac{\partial u_k}{\partial x_k} \tag{2.5}$$

here

μ = first coefficient of viscosity for the mixture

λ = second coefficient of viscosity for the mixture

f_{s_i} = i^{th} component of the external force acting on the s^{th} species

δ_{ij} = Kronecker delta

Y_s = mass fraction of the s^{th} species.

The first coefficient of viscosity is also known as the dynamic or shear viscosity coefficient. Sometimes the bulk viscosity coefficient, β, is introduced instead of the second coefficient, λ, which is given by

$$\beta = \lambda + \frac{2}{3}\mu. \tag{2.6}$$

Note that *Stoke's hypothesis* yields

$$(2\mu + 3\lambda) \sum_{k=1}^{3} \frac{\partial u_k}{\partial x_k} = \sum_{i=1}^{3} \tau_{ii} = 0 \tag{2.7}$$

which implies that for a compressible gaseous mixture

$$\lambda = -\frac{2}{3}\mu \quad \text{or} \quad \beta = 0. \tag{2.8}$$

However, this generally only holds for mono-atomic gaseous mixtures. Note that the stress tensor $\bar{\bar{\tau}}$ can be written in the following generalized form

$$\bar{\bar{\tau}} = \mu\left[\nabla\mathbf{V} + (\nabla\mathbf{V})^T\right] + \lambda(\nabla\cdot\mathbf{V})\,\bar{\bar{I}} \tag{2.9}$$

where $\bar{\bar{I}}$ is a unit tensor and the superscript T denotes the transposition operation.

The momentum equation in general tensor form is

$$\frac{\partial}{\partial t}(\rho\mathbf{V}) + \nabla\cdot(\rho\mathbf{V}\circ\mathbf{V}) + \nabla p = \nabla\cdot\bar{\bar{\tau}} + \sum_{s=1}^{S}\mathbf{F}_s\rho Y_s \tag{2.10}$$

where the symbol $\circ$ implies the dyadic tensor operation. The generalized vector momentum equation can be written as

$$\begin{aligned}\rho\left[\frac{\partial\mathbf{V}}{\partial t} - \mathbf{V}\times(\nabla\times\mathbf{V}) + \frac{1}{2}\nabla V^2\right] = {} & \sum_{s=1}^{S}\mathbf{F}_s\rho Y_s - \nabla p + \nabla(\lambda\nabla\cdot\mathbf{V}) + \\ & \mu\nabla(\nabla\cdot\mathbf{V}) + (\nabla\mu\cdot\nabla)\mathbf{V} + \mu\nabla^2\mathbf{V} + \\ & \nabla(\mathbf{V}\cdot\nabla\mu) - (\mathbf{V}\cdot\nabla)\nabla\mu.\end{aligned} \tag{2.11}$$

2.2.3 Species Equations

The rate of change of mass fraction Y_s of the s^{th} species in a system at any time is equal to the sum of three terms: (1) the influx of species into the system due to advection, (2) the net rate of production of the species due to chemical reactions, and (3) the net diffusional *influx* of this species into the system. The species equations in vector form are

$$\frac{\partial}{\partial t}(\rho Y_s) + \nabla \cdot (\rho Y_s \mathbf{V}) \;=\; \dot{W}_s - \nabla \cdot (\rho Y_s \mathbf{V}_s) \qquad s = 1, \ldots, S \tag{2.12}$$

where $\dot{W}_s$ is the net mass rate of production of the s^{th} species per unit volume due to *all* of the chemical reactions and $\mathbf{V}_s$ is its diffusional velocity. This equation, in indicial form, becomes

$$\frac{\partial}{\partial t}(\rho Y_s) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(\rho u_j Y_s + \rho V_{s_j} Y_s\right) \;=\; \dot{W}_s \;, \qquad s = 1, \ldots, S. \tag{2.13}$$

The mass production rate $\dot{W}_s$ is related to the molal production rate $\dot{w}_s$ by

$$\dot{W}_s \;=\; \hat{m}_s \dot{w}_s \tag{2.14}$$

where $\hat{m}_s$ is the molecular mass of species s. Since mass is conserved in each separate reaction we have

$$\sum_{s=1}^{S} (\beta_{sr} - \alpha_{sr}) \hat{m}_s \;=\; 0 \;, \qquad r = 1, \ldots, R. \tag{2.15}$$

Note that Equations (2.13) are not mutually independent. That is the sum of all S equations results in the continuity equation, since

$$\sum_{s=1}^{S} Y_s \;=\; 1 \tag{2.16}$$

and

$$\sum_{s=1}^{S} \dot{W}_s \;=\; \sum_{s=1}^{S} \hat{m}_s \dot{w}_s \;=\; 0. \tag{2.17}$$

The last equation expresses the fact that mass is neither created nor destroyed due to chemical reactions if nuclear transformations are excluded. The fact that summation of

the diffusional mass fluxes over all species, with respect to an observer moving at the *local mass average velocity* or *barycentric velocity*, must be zero translates into

$$\sum_{s=1}^{S} \mathbf{J}_s = \sum_{s=1}^{S} Y_s \mathbf{V}_s = 0. \tag{2.18}$$

The diffusional velocity $\mathbf{V}_s$ is given by the so-called Fick's law. For multicomponent gaseous mixtures the diffusional law becomes very complex because the diffusion flux $\mathbf{J}_s$ of each species depends upon the concentration gradients of all components in the mixture. There are additional effects due to pressure gradients (when mass fraction differs from mole fraction), temperature gradients (Soret effect) and differences in body forces on molecules of different species [141]. However an approximate expression which neglects coupled effects and lumps the multicomponent contribution is generally used as a constitutive relation [131]

$$\mathbf{J}_s = \rho Y_s \mathbf{V}_s \approx -\rho D_s \nabla Y_s \tag{2.19}$$

where the diffusion coefficient D_s of species s is

$$D_s = \frac{1 - Y_s}{\sum_{\substack{j=1 \\ j \neq s}}^{S} \frac{\hat{m} Y_j}{\hat{m}_j D_{sj}}} \tag{2.20}$$

where $\hat{m}$ is the molecular mass of the mixture which is given by

$$\frac{1}{\hat{m}} = \sum_{s=1}^{S} \frac{Y_s}{\hat{m}_s} \tag{2.21}$$

D_{sj} is the binary diffusion coefficient for species s and i. Substituting Equation (2.19) into Equations (2.13) results in

$$\frac{\partial}{\partial t}(\rho Y_s) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j} \left[\left(\rho u_j - \rho D_s \frac{\partial Y_s}{\partial x_j} \right) Y_s \right] = \dot{W}_s \ , \qquad s = 1, \ldots, S. \tag{2.22}$$

The species production rate is given by the following non-linear phenomenological chemical kinetic expression

$$\dot{w}_s = \sum_{r=1}^{R} (\beta_{sr} - \alpha_{sr}) \left[K_{fr} \prod_{l=1}^{S} \left(\frac{\rho Y_l}{\hat{m}_l} \right)^{\alpha'_{lr}} - K_{br} \prod_{l=1}^{S} \left(\frac{\rho Y_l}{\hat{m}_l} \right)^{\beta'_{lr}} \right] \tag{2.23}$$

where K_{fr}, K_{br} are forward, backward rate coefficients for reaction r and the exponents α'_{lr}, β'_{lr} specify the order of this reaction for species l. For elementary reactions $\alpha'_{lr} = \alpha_{lr}$

and $\beta'_{lr} = \beta_{lr}$. In an attempt to reduce the total number of reactions, a chemical reaction system is sometimes replaced by a single, one step irreversible reaction. For such complete reactions, the *order* of reaction is often different from the *molecularity* and the second term on the right hand side of Equation (2.23) is disregarded in calculating the contributions to species production rate. For ease of understanding one frequently defines the *progress rate* of a reaction as

$$\Omega_l = K_l \prod_{s=1}^{S} \left(\frac{\rho Y_s}{\hat{m}_s}\right)^{\sigma} \tag{2.24}$$

where $\sigma = \alpha'_{sr}, \beta'_{sr}$ for $l = fr, br$. Then the mass production rate of species s becomes

$$\dot{W}_s = \hat{m}_s \sum_{r=1}^{R} (\beta_{sr} - \alpha_{sr}) (\Omega_{fr} - \Omega_{br}) . \tag{2.25}$$

The quantity $C_s = \rho Y_s / \hat{m}_s$ is frequently known as the *concentration* of species s. The rate constants are assumed to be of the generalized Arrhenius form

$$K_{fr} = A_{fr} T^{\eta_{fr}} \exp(-E_{fr}/\mathcal{R}T) \tag{2.26}$$

$$K_{br} = K_{fr}/K_{cr} \tag{2.27}$$

where K_{cr} is the equilibrium constant for reaction r. These expressions implicitly assume that all internal degrees of freedom (rotation, vibration, electronic excitation) are in equilibrium with the translational mode, *i.e.*, a single temperature is assumed for all internal degrees of freedom. For most species (except near cryogenic temperatures) the rotational mode is in equilibrium with the translational one. At temperatures of order 10^3 K the vibrational modes of most species are not in equilibrium with the translational modes and the above single temperature model becomes unreliable. Park [103] has recently advocated a two-temperature thermo-chemical model which recognizes the dependence of rate processes on both translational and vibrational temperatures. He also assumes that electron temperature and electronic excitation temperature are close to the vibrational temperature, and that rate constants are dictated by a geometric mean temperature between the translational and vibrational temperatures. This also means that an additional partial differential equation has to be solved for the vibrational temperature.

To extract the dimensions of K_{fr} consider a unidirectional single reaction with advection terms dropped, then the species rate is governed by

$$\frac{dC_s}{dt} = (\beta_{sr} - \alpha_{sr}) K_{fr} \prod_{l=1}^{S} C_s^{\alpha'_{lr}}.$$

Hence the dimensions of K_{fr} are $(mole/volume)^{(1-z_f)}\, time^{-1}$ where $z_f = \sum_{s=1}^{S} \alpha'_{sr}$. In a similar manner the dimensions of K_{br} are $(mole/volume)^{(1-z_b)}\, time^{-1}$ where $z_b = \sum_s \beta'_{sr}$, and the dimensions of K_{cr} are $(mole/volume)^{z_c}$ where $z_c = \sum_s (\beta_{sr} - \alpha_{sr})$. The procedure for the determination of equilibrium constants will be discussed later in this chapter.

2.2.4 Energy Equation

The total specific internal energy, E, of the mixture is defined as the sum of specific internal and specific kinetic energies of the mixture

$$E = h - \frac{p}{\rho} + \frac{1}{2} V^2 \tag{2.28}$$

where h is the specific enthalpy of the mixture. The conservation of the total specific internal energy is governed by

$$\begin{aligned} \frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\rho E \mathbf{V}) &= -\nabla \cdot \mathbf{q} - \nabla \cdot (p\mathbf{V}) + \nabla \cdot (\mathbf{V} \cdot \bar{\bar{\tau}}) \\ &\quad + \mathbf{V} \cdot \sum_{s=1}^{S} \rho Y_s \mathbf{F}_s + \sum_{s=1}^{S} \rho Y_s \mathbf{V}_s \cdot \mathbf{F}_s \end{aligned} \tag{2.29}$$

where $\nabla \cdot \mathbf{q}$ represents the overall heat flux which has contributions from (1) external heat conduction, (2) heat radiation flux, (3) energy flux due to species diffusion and (4) thermal diffusion flux. The external heat conduction flux is given by the Fourier law. For a multicomponent fluid of non-uniform composition there are additional contributions from the energy flux due to diffusion of various species with different enthalpies and the coupled effects between transfers of mass and energy *i.e.*, Dufour effect [141]. Neglecting the coupled effect, which is usually small compared to the direct effects, the following phenomenological expression for the overall heat flux can be obtained

$$\mathbf{q} = -k \nabla T + \rho \sum_{s=1}^{S} h_s Y_s \mathbf{V}_s + \mathbf{q}_r \tag{2.30}$$

where k is the coefficient of thermal conductivity for the mixture, q_r is the radiation heat flux and h_s is the specific enthalpy of the species s which is given by

$$h_s = H_{f_s} + \int_{T_0}^{T} C_{p_s} dT. \tag{2.31}$$

Here H_{f_s} is the standard specific heat of formation for species s at the reference temperature T_0 and C_{p_s} is its constant pressure specific heat.

Defining $\epsilon = \rho E$ to be the total internal energy per unit volume, the indicial tensor form of the energy equation becomes

$$\frac{\partial \epsilon}{\partial t} + \sum_{i=1}^{3} \frac{\partial}{\partial x_i} \left[(\epsilon + p) u_i - \sum_{j=1}^{3} u_j \tau_{ij} + q_i \right] = \rho \sum_{s=1}^{S} Y_s \sum_{i=1}^{3} (u_i + V_{s_i}) f_{s_i} \tag{2.32}$$

with the following two constitutive relations. First, using Equations (2.28) and (2.31), the caloric equation of state becomes

$$\frac{\epsilon}{\rho} = \sum_{s=1}^{S} Y_s \left\{ H_{f_s} + \int_{T_0}^{T} C_{p_s} dT \right\} + \sum_{j=1}^{3} \frac{u_j^2}{2} - \frac{p}{\rho} \tag{2.33}$$

and second, using Equations (2.30), (2.31) and (2.19), the heat flux components can be written as

$$q_i = -k \frac{\partial T}{\partial x_i} - \rho \sum_{s=1}^{S} \left\{ H_{f_s} + \int_{T_0}^{T} C_{p_s} dT \right\} D_s \frac{\partial Y_s}{\partial x_i} + q_{r_i} \qquad i = 1, \ldots, 3. \tag{2.34}$$

2.2.5 Thermal Equation of State and Constitutive Relations

The conservation equations are supplemented by one or more constitutive relations which express the relationship between state properties and transport coefficients. The relationship describing the variation of temperature, pressure and density is referred to as thermal equation of state. Since each gas component is assumed to be a perfect gas satisfying Dalton's law of partial pressures, the equation of state for the mixture becomes

$$\frac{p}{\rho} = \mathcal{R} T \sum_{s=1}^{S} \frac{Y_s}{\hat{m}_s}. \tag{2.35}$$

The other constitutive relations pertain to the models for coefficients governing the diffusion of momentum, energy and species. The individual species dynamic viscosities

can be determined from the Sutherland approximation which results from the kinetic theory using an idealized inter-molecular force potential and is as follows:

$$\frac{\mu_s}{\mu_{s_0}} = \left(\frac{T}{T_0}\right)^{1.5} \frac{T_0 + S_s}{T + S_s} \tag{2.36}$$

where μ_{s_0} and T_0 are reference values and S_s is the Sutherland constant. These reference values are tabulated for some species in References [124,138]. The mixture viscosity can be determined from Wilke's formula, [140]

$$\mu = \sum_{s=1}^{S} \frac{Y_s \mu_s}{\sum_{j=1}^{S} \frac{\hat{m}_s}{\hat{m}_j} Y_j \phi_{sj}} \tag{2.37}$$

where

$$\phi_{sj} = \frac{\left\{1 + \left(\frac{\mu_s Y_j}{\mu_j Y_s}\right)^{0.5} \left(\frac{\hat{m}_s}{\hat{m}_j}\right)^{0.25}\right\}^2}{\left[8\left(1 + \frac{\hat{m}_s}{\hat{m}_j}\right)\right]^{0.5}}. \tag{2.38}$$

The individual species thermal conductivities can also be computed from the Sutherland law

$$\frac{k_s}{k_{s_0}} = \left(\frac{T}{T_0}\right)^{1.5} \frac{T_0 + \acute{S}_s}{T + \acute{S}_s} \tag{2.39}$$

where k_{s_0}, T_0 and $\acute{S}_s$ are constants. These values are also tabulated in References [124,138]. The mixture thermal conductivity can be determined from the following formula, [43]

$$k = \sum_{s=1}^{S} \frac{Y_s k_s}{\sum_{j=1}^{S} \frac{\hat{m}_s}{\hat{m}_j} Y_j \acute{\phi}_{sj}} \tag{2.40}$$

where $\acute{\phi}_{sj}$ is related to ϕ_{sj} by

$$\acute{\phi}_{sj} = \begin{cases} 1.065\phi_{sj} & \text{if } s \neq j \\ 1.0 & \text{otherwise.} \end{cases} \tag{2.41}$$

Chapman and Cowling used kinetic theory of dilute gases to arrive at the following expression for binary diffusion coefficient D_{sj} between species s and j, [138]

$$D_{sj} = 0.1858 \times 10^{-6} \frac{T^{1.5} \left[(\hat{m}_s + \hat{m}_j)/\hat{m}_s \hat{m}_j\right]^{0.5}}{p \sigma_{sj}^2 \Omega_D} \qquad m^2/s \tag{2.42}$$

where T is the mixture temperature in degree Kelvin, p is the mixture pressure in atmospheres, the effective collision diameter σ_{sj} is in Ångstrom units (Å) and Ω_D is the

dimensionless collision integral which can be approximated by

$$\Omega_D = \left(\frac{T^{\epsilon}_{sj}}{T}\right)^{0.145} + \left(\frac{2T^{\epsilon}_{sj}}{2T + T^{\epsilon}_{sj}}\right)^2 . \tag{2.43}$$

The effective temperatures T^{ϵ}_{sj} and diameters σ_{sj} are averages computed from individual molecular properties, *viz.*,

$$\begin{aligned} \sigma_{sj} &= 0.5(\sigma_s + \sigma_j) \\ T^{\epsilon}_{sj} &= \left(T^{\epsilon}_s T^{\epsilon}_j\right)^{0.5} . \end{aligned} \tag{2.44}$$

The values of the effective temperatures and diameters are tabulated in Reference [138] for some gases. Once the binary diffusion coefficients for all species combinations are known, the species diffusion coefficients D_s can be computed from the approximate formula of Equation (2.20).

This completes the set of governing equations and the constitutive relations. This is an extremely rich set of equations. Combined with appropriate initial and boundary conditions, these equations describe such interesting phenomena as flames, detonations, combustion noise and instabilities, smoldering fires, shock tubes flows, turbulence, *etc.* They are, in fact, sufficiently difficult to solve that entire disciplines have been devoted to solving only subsets of them for specific applications. The difficulties encountered in solving these equations stem from physical, computational and mathematical problems. The input parameters, such as rate or diffusion coefficients, are either not known or there exist vast discrepancies in the experimentally observed values. The other issue which pertains to the understanding of physics is the inadequate treatment of the turbulence phenomenon. The computational problems involve inadequate numerical methods to resolve physical phenomena, insufficient computer memory, and prohibitively long CPU time. The mathematical problems relate to stiffness introduced due to widely disparate time scales.

In subsequent sections the effects of viscosity, diffusion, heat transfer and external forces will be neglected, and consideration will be limited to either one or two spatial dimensions. The thrust of the present study is the development of an adaptive algorithm for applications involving unsteady inviscid (Euler) flows.

2.3 Quasi 1-D Inviscid Equations

The governing conservation equations for a one-dimensional streamtube can be written in the compact form

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} = W. \tag{2.45}$$

Here

$$U = \begin{pmatrix} \rho A \\ \rho A u \\ A\epsilon \\ \rho A Y_s \end{pmatrix}, \qquad F = \begin{pmatrix} \rho A u \\ A(\rho u^2 + p) \\ Au(\epsilon + p) \\ \rho A u Y_s \end{pmatrix}, \qquad W = \begin{pmatrix} 0 \\ p dA/dx \\ 0 \\ A\hat{m}_s \dot{w}_s \end{pmatrix} \tag{2.46}$$

where A is the stream-tube area. The fourth entry in these vectors corresponds to $s = 1, \cdots, S-1$, where one of the species equations has been omitted in favor of the global continuity equation. The source terms $\dot{w}_s$ are given by Equation (2.23). The normalization of the quasi-one-dimensional Euler equations is similar to that for the two-dimensional equations to be discussed next.

2.4 2-D Inviscid Equations

The compact form of the two-dimensional Euler equations is

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = W. \tag{2.47}$$

Here

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \epsilon \\ \rho Y_s \end{pmatrix}, \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\epsilon + p)u \\ \rho u Y_s \end{pmatrix}, \quad G = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\epsilon + p)v \\ \rho v Y_s \end{pmatrix}, \quad W = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dot{W}_s \end{pmatrix}. \tag{2.48}$$

Again the fifth entry in these vectors corresponds to $s = 1, \cdots, S-1$.

2.4.1 Normalization

Let the subscript r indicate some reference conditions and denote the non-dimensional quantities by asterisks, *i.e.*, define

$$\begin{aligned}
x &= x^* L_r & y &= y^* L_r & t &= t^* t_r \\
\rho &= \rho^* \rho_r & u &= u^* u_r & v &= v^* u_r \\
p &= p^* p_r & T &= T^* T_r & \dot{W}_s &= \dot{W}_s^* \dot{W}_r \\
\epsilon &= \epsilon^* \epsilon_r & C_{p_s} &= C_{p_s}^* C_{p_r} & Y_s &= Y_s^* \\
H_{f_s} &= H_{f_s}^* H_r & \hat{m}_s &= \hat{m}_s^* \hat{m}_r.
\end{aligned} \tag{2.49}$$

In order to keep the form of the dimensional and normalized equations invariant, the continuity equation dictates

$$t_r = L_r / u_r. \tag{2.50}$$

The momentum equations yield

$$u_r^2 = p_r / \rho_r \tag{2.51}$$

whereas the energy equation yields

$$\epsilon_r = p_r = \rho_r u_r^2. \tag{2.52}$$

The species equations yield

$$\dot{W}_r = \rho_r u_r / L_r. \tag{2.53}$$

The rate coefficients K_{f_r}, K_{b_r} in Equations (2.26) and (2.27) are usually given in dimensional units which vary from one reaction to another. For this reason the mass production rates in the STAR code are first computed in dimensional form by using Equation (2.25) and are subsequently normalized by the factor $\dot{W}_r = \rho_r u_r / L_r$.

The form of the caloric equation of state (Eq. 2.33) is kept invariant by the choice

$$H_r = C_{p_r} T_r = u_r^2 \tag{2.54}$$

so the non-dimensional definition of specific total energy is (dropping asterisks)

$$\frac{\epsilon}{\rho} = \sum_{s=1}^{S} Y_s \left\{ H_{f_s} + \int_{T_0}^{T} C_{p_s} dT \right\} + \frac{u^2 + v^2}{2} - \frac{p}{\rho}. \tag{2.55}$$

The thermal equation of state yields

$$p^* p_r = \rho^* \rho_r \mathcal{R} T_r T^* \sum_{s=1}^{S} \frac{Y_s}{\hat{m}_s^* \hat{m}_r}. \tag{2.56}$$

Thus the thermal equation of state in normalized form becomes

$$p^* = \rho^* T^* \sum_{s=1}^{S} \frac{Y_s}{\hat{m}_s^*} \tag{2.57}$$

in which it is natural to choose

$$p_r = \frac{\rho_r \mathcal{R} T_r}{\hat{m}_r}. \tag{2.58}$$

Since the mass fraction is already a dimensionless quantity, it is the same in both dimensional and non-dimensional equations. In all later computations $\hat{m}_r$ was chosen to be the molecular mass of the gaseous mixture at the reference state; this implies that if the mass fraction of species s at reference state is denoted by Y_{s_r}, then Equation (2.21) yields

$$\hat{m}_r = \frac{1}{\sum_{s=1}^{S} \frac{Y_{s_r}}{\hat{m}_s}}. \tag{2.59}$$

Note that the thermal equation of state is the only one which is slightly modified in non-dimensional form. Henceforth the non-dimensional equations will be written with the asterisks omitted.

2.5 Inviscid Equations in Transformed Coordinates

The algorithm for a set of partial differential equations can be made appreciably more robust by utilizing a well-constructed grid. It is well-known that an improper choice of node point locations can lead to unsatisfactory results or instabilities in extreme cases. However, the choice of grids in most cases is dictated by the boundaries of the physical domain, or by the presence of large solution gradients in certain spatial locations. Thus the cell volumes in physical coordinates often differ; in addition these cells may be highly skewed or compressed in a single direction. One can remove such non-uniformity by utilizing mappings to transform the physical domain into a uniform computational domain. Thus the governing equations in physical coordinates, (t, x, y), in general,

are transformed into an appropriate computational domain (τ, ξ, η) for solution. The mapping need not be globally one-to-one but must be so locally.

The generalized coordinate mapping in this study is time-invariant (*i.e.*, the grid is stationary while the integration is being performed) and hence is of the form

$$\tau = t, \qquad \xi = \xi(x,y), \qquad \eta = \eta(x,y). \tag{2.60}$$

The notation of "Jacobian Algebra" will be used here to derive the transformed equations. Note the Equations (2.47) can be written in the form

$$\frac{\partial U}{\partial t} + \frac{\partial(F,y)}{\partial(x,y)} + \frac{\partial(x,G)}{\partial(x,y)} = W. \tag{2.61}$$

The Jacobian of the transformation is

$$J = \frac{\partial(\xi,\eta)}{\partial(x,y)} = \begin{vmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{vmatrix} = \xi_x\eta_y - \eta_x\xi_y \tag{2.62}$$

where for example $(.)_x$ denotes differentiation with respect to x. In two spatial dimensions, the Jacobian of the transformation controls the magnification of area elements between the physical and computational domains. For the transformation to be locally one-to-one the Jacobian of the transformation must be finite and not vanish.

With the help of the previous two equations the conservation law can be rewritten as

$$\frac{\partial(x,y)}{\partial(\xi,\eta)}\frac{\partial U}{\partial t} + \frac{\partial(F,y)}{\partial(\xi,\eta)} + \frac{\partial(x,G)}{\partial(\xi,\eta)} = \frac{\partial(x,y)}{\partial(\xi,\eta)}W \tag{2.63}$$

or

$$\frac{1}{J}\frac{\partial U}{\partial t} + [y_\eta F_\xi - y_\xi F_\eta + x_\xi G_\eta - x_\eta G_\xi] = \frac{1}{J}W. \tag{2.64}$$

Noting that the sum of the following identities

$$\begin{aligned} \tfrac{\partial}{\partial\xi}(y_\eta F - x_\eta G) &= y_\eta F_\xi + F y_{\eta\xi} - G x_{\eta\xi} - x_\eta G_\xi \\ \tfrac{\partial}{\partial\eta}(x_\xi G - y_\xi F) &= x_\xi G_\eta + G x_{\eta\xi} - F y_{\eta\xi} - y_\xi F_\eta \end{aligned} \tag{2.65}$$

is the square bracket in Equation (2.64), the conservation equations for two-dimensional unsteady reacting flow in general curvilinear coordinates can now be written in compact form as

$$\frac{\partial \tilde{U}}{\partial t} + \frac{\partial \tilde{F}}{\partial \xi} + \frac{\partial \tilde{G}}{\partial \eta} = \tilde{W} \tag{2.66}$$

where the state vector $\tilde{U}$, flux vectors $\tilde{F}, \tilde{G}$ and the source vector $\tilde{W}$ in the curvilinear coordinates can be related to the corresponding Cartesian vectors by

$$\begin{aligned} \tilde{U} &= U/J \\ \tilde{F} &= y_\eta F - x_\eta G \\ \tilde{G} &= x_\xi G - y_\xi F \\ \tilde{W} &= W/J. \end{aligned} \tag{2.67}$$

The quantities $(x_\xi, x_\eta, y_\xi, y_\eta)$ are referred to as the transformation metrics which can be computed once the physical grid is specified.

2.6 Primitive Variables

After obtaining the state variables $(\rho, \rho u, \rho v, \epsilon, \rho Y_s)$ at a new time level, the primitive variables $(\rho, u, v, \epsilon, Y_s, p, T)$ may have to be evaluated. From the definition of the state vector U in Equation (2.48) it can be seen that some of the primitive variables can be obtained by simply dividing the components of state vector by the density. However, the decoding of temperature and pressure is non-trivial due to the complexity of the caloric equation of state. Over a given range of temperatures it is reasonable to assume that the constant pressure specific heat for each species is a linear function of temperature, *i.e.,*

$$C_{p_s}(T) = a_s + b_s T \quad , \quad s = 1, \ldots, S \tag{2.68}$$

where a_s and b_s are constants [44]. The following procedure for evaluation of temperature pertains to this thermodynamic model. The caloric equation of state (Eq. 2.55) can now be integrated and written in the following form

$$\frac{\epsilon}{\rho} - \sum_{s=1}^{S} Y_s H_{f_s} - \frac{u^2+v^2}{2} = (T - T_0)\sum_{s=1}^{S} Y_s a_s + \frac{1}{2}(T^2 - T_0^2)\sum_{s=1}^{S} Y_s b_s - \mathcal{R}T \sum_{s=1}^{S} \frac{Y_s}{\hat{m}_s} \tag{2.69}$$

or

$$\frac{1}{2} A_m T^2 + B_m T = C_m \tag{2.70}$$

where

$$\begin{aligned} A_m &= \sum_{s=1}^{S} Y_s b_s \\ B_m &= \sum_{s=1}^{S} Y_s \left[a_s - \frac{\mathcal{R}}{\hat{m}_s} \right] \\ C_m &= \frac{\epsilon}{\rho} - \sum_{s=1}^{S} Y_s H_{f_s} - \frac{u^2+v^2}{2} + T_0 \sum_{s=1}^{S} Y_s a_s + \frac{1}{2} A_m T_0^2. \end{aligned} \tag{2.71}$$

Note that A_m, B_m, C_m involve only the primitive variables which are already decoded. Solving the quadratic equation for T and selecting only the meaningful positive root yields

$$T = \frac{2C_m}{\sqrt{B_m^2 + 2A_m C_m} + B_m}. \tag{2.72}$$

The situation $A_m = 0$ occurs for calorically perfect mixture (constant C_p for each species) in which case $B_m = \sum Y_s C_{v_s}$ and hence

$$T = \frac{\frac{\epsilon}{\rho} - \sum Y_s H_{f_s} - \frac{u^2+v^2}{2} + T_0 \sum Y_s C_{p_s}}{\sum Y_s C_{v_s}}. \tag{2.73}$$

Once the temperature is known the pressure can be obtained from the thermal equation of state.

2.7 Equilibrium Rate Constants

Consider a closed system containing a mixture of reacting perfect gases with a fixed temperature T and pressure p. The degree of reaction λ_r of a specific reaction r is given by

$$dn_s|_r = (\beta_{sr} - \alpha_{sr}) d\lambda_r \tag{2.74}$$

where n_s denotes the number of moles of species s and $dn_s|_r$ denotes the change of this number due to reaction r [134]. This equation states that the change in the number of moles follows stoichiometric proportions. For example, for the reaction, $H_2 + O_2 \rightleftharpoons 2OH$ a depletion of 2 moles of H_2 would mean a corresponding depletion of 2 moles of O_2 and a formation of 4 moles of OH.

Since the entropy of an ideal gas is governed by

$$ds = C_p \frac{dT}{T} - \frac{\mathcal{R}}{\hat{m}} \frac{dp}{p} \tag{2.75}$$

the Gibbs free energy for a constituent is

$$g_s = h_s - Ts_s = H_{f_s} + \int_{T_0}^{T} C_{p_s} dT - Ts_0 - T\int_{T_0}^{T} C_{p_s}\frac{dT}{T} + \frac{\mathcal{R}T}{\hat{m}_s}\int_{p_0}^{p_s}\frac{dp_s}{p_s} \quad (2.76)$$

where s_0 refers to the absolute entropy at the standard temperature T_0 (usually 273 K) and pressure p_0 (usually 0.1 MPa). Also note that the pressure in Equation (2.75) is replaced by the partial pressure of species s and the pressure integral is evaluated from the ***pure state*** pressure p_0 to the current partial pressure of the constituent [77]. The specific species Gibbs function can be rewritten as

$$g_s = \omega_s(T) + \frac{\mathcal{R}T}{\hat{m}_s}\ln(\frac{p_s}{p_0}) \quad (2.77)$$

where

$$\omega_s(T) = H_{f_s} - Ts_0 + \int_{T_0}^{T} C_{p_s} dT - T\int_{T_0}^{T} C_{p_s}\frac{dT}{T}. \quad (2.78)$$

The total Gibbs free energy of the mixture is given by

$$G = \sum_{s=1}^{S} n_s \bar{g}_s = \sum_{s=1}^{S} n_s \hat{m}_s g_s \quad (2.79)$$

where $\bar{g}_s$ is the partial molal Gibbs function which for perfect gases is also the ***chemical potential.*** Equilibrium is attained for the system if the Gibbs free energy G achieves a minimum [77]. Furthermore, constancy of temperature and pressure is a precondition for thermal and mechanical equilibria. These conditions imply

$$dG_{p,T} = d\left(\sum_{s=1}^{S} n_s \hat{m}_s g_s\right) = \sum_{s=1}^{S} \hat{m}_s \left(n_s dg_s + g_s dn_s\right) = 0. \quad (2.80)$$

For simultaneously occurring multiple reactions

$$dn_s = \sum_{r=1}^{R} dn_s|_r = \sum_{r=1}^{R} (\beta_{sr} - \alpha_{sr}) d\lambda_r. \quad (2.81)$$

Furthermore since T is held constant, Equation (2.77) implies

$$dg_s = \frac{\mathcal{R}T}{\hat{m}_s}\frac{dp_s}{p_s}. \quad (2.82)$$

Substituting Equations (2.81) and (2.82) in Equation (2.80) yields

$$\sum_{s=1}^{S} n_s \mathcal{R}T\frac{dp_s}{p_s} + \sum_{s=1}^{S} \hat{m}_s g_s \sum_{r=1}^{R} (\beta_{sr} - \alpha_{sr}) d\lambda_r = 0. \quad (2.83)$$

The first of these terms is zero since pressure is constant and $p_s V = n_s \mathcal{R} T$, hence

$$\sum_{s=1}^{S} n_s \mathcal{R} T \frac{dp_s}{p_s} = \sum_{s=1}^{S} V dp_s = V \sum_{s=1}^{S} dp_s = V d(\sum_{s=1}^{S} p_s) = V dp = 0.$$

Thus Equation (2.83) implies

$$\sum_{r=1}^{R} \sum_{s=1}^{S} \hat{m}_s (\beta_{sr} - \alpha_{sr}) \left[\omega_s(T) + \frac{\mathcal{R} T}{\hat{m}_s} \ln(\frac{p_s}{p_0}) \right] d\lambda_r = 0. \tag{2.84}$$

Since the reaction r can also occur separately, each of the $d\lambda_r$ may be varied independently and hence

$$K_{p_r} = \exp(\frac{-\Delta G_r}{\mathcal{R} T}) = \prod_{s=1}^{S} \left(\frac{p_s}{p_0} \right)^{(\beta_{sr} - \alpha_{sr})} \tag{2.85}$$

where K_{p_r} is known as the equilibrium constant for partial pressures and ΔG_r is given by

$$\Delta G_r = \sum_{s=1}^{S} (\beta_{sr} - \alpha_{sr}) \omega_s(T). \tag{2.86}$$

The equilibrium constant for concentrations is given by

$$K_{c_r} = \prod_{s=1}^{S} \left(\frac{\rho Y_s}{\hat{m}_s} \right)^{(\beta_{sr} - \alpha_{sr})} = \prod_{s=1}^{S} \left(\frac{p_s}{\mathcal{R} T} \right)^{(\beta_{sr} - \alpha_{sr})}. \tag{2.87}$$

Substitution of Equation (2.85) in (2.87) results in

$$K_{c_r} = K_{p_r} \prod_{s=1}^{S} \left(\frac{p_0}{\mathcal{R} T} \right)^{(\beta_{sr} - \alpha_{sr})} = K_{p_r} \left(\frac{p_0}{\mathcal{R} T} \right)^{\sum_{s=1}^{S} (\beta_{sr} - \alpha_{sr})}. \tag{2.88}$$

Note that K_{p_r} is a dimensionless quantity whereas K_{c_r} has the dimensions $kmole/m^3$ raised to the power $\sum(\beta_{sr} - \alpha_{sr})$. As has been shown here both the equilibrium constants depend only on the temperature for a mixture of ideal gases.

For an accurate description of the reaction system, the equilibrium constants must be determined by the above procedure (Eqs. 2.78, 2.85, 2.86, and 2.88) at all the spatial locations and at each time-level. Consider that for a typical 100×50 grid with two reactions and 1000 time-steps the above calculations must be repeated 10^7 times. A simpler model for the equilibrium constants can lead to substantial savings. For engineering purposes the equilibrium constant is usually approximated over a given range of temperatures by the following expression [136]

$$K_{cr} = A_{cr} T^{\eta_{cr}} \exp(-E_{cr}/\mathcal{R} T). \tag{2.89}$$

This is consistent with forward and backward rate coefficient forms in Equations (2.26) and (2.27) being written in the generalized Arrhenius form. For the STAR code the constants $A_{cr}, \eta_{cr}, E_{cr}$ can either be user-supplied or can be calculated to match K_{cr} at three representative temperatures. The equilibrium constants are determined by the longer procedure at temperatures T_1, T_2, T_3 and their values are denoted by $K_{c_1}, K_{c_2}, K_{c_3}$ respectively. The following system of linear equations is then solved to determine the unknown constants [132].

$$\begin{pmatrix} 1 & \ln T_1 & -\frac{1}{\mathcal{R}T_1} \\ 1 & \ln T_2 & -\frac{1}{\mathcal{R}T_2} \\ 1 & \ln T_3 & -\frac{1}{\mathcal{R}T_3} \end{pmatrix} \begin{pmatrix} \ln A_c \\ \eta_c \\ E_c \end{pmatrix} = \begin{pmatrix} \ln K_{c_1} \\ \ln K_{c_2} \\ \ln K_{c_3} \end{pmatrix}. \tag{2.90}$$

Different choices of temperatures yields different values of constants, but the numerical value of the rate constants differ only slightly. Frequently the range of temperatures is known *apriori* and this knowledge can be used to choose appropriate values of T_1, T_2, T_3.

2.8 Chemistry Reaction Models

Two chemical models have been considered in this study. The first describes dissociation-recombination in terms of a Lighthill ideal gas. This model was used to examine the potential difficulties encountered in the spatio-temporal algorithm. The second model, describes hydrogen-air combustion and was used to demonstrate the applicability of the developed algorithm to multi-component, multi-reaction systems.

2.8.1 Lighthill Dissociation Model

In 1957 Lighthill [78] proposed a simplified model to describe a dissociating gas flow in equilibrium and referred to it as an ideal dissociating gas. A year later Freeman [52] used the model to describe non-equilibrium situations. Denoting the atom by the

chemical symbol Z the dissociating reaction is

$$Z_2 \rightleftharpoons 2Z. \tag{2.91}$$

For the model to be applicable to real gases, the temperature range for the flow should be such that dissociation occurs appreciably but ionization is negligible. For gases like O_2 and N_2 the approximate temperature range is 1000 to 7000 K. The Lighthill model assumes vibrational modes to be excited to one half the maximum classical value. At relatively high temperatures the actual molecular excitation may be more than the factor of one half, but the molecules themselves are reduced in number due to dissociation and in the process absorb energy thereby compensating for the underestimation of vibrational levels. The *frozen* ratio of specific heats may be written as

$$\gamma_f = \frac{4+Y_Z}{3}. \tag{2.92}$$

At low temperatures when $Y_Z \approx 0$, the ideal dissociating gas is a perfect gas with constant specific heats and $\gamma_f = 4/3$. The difference from 7/5 is a result of the assumption that the vibrational degrees of freedom are one half excited even at low temperatures. Hence for this model to be a realistic match to air the lower temperature limit is about 1000 K. Note that for O_2 at 1000 K there is no appreciable dissociation and the ratio of specific heats is 1.31.

The species for this model are numbered as

$$Y_1 = Y_Z \qquad Y_2 = Y_{Z_2} = 1 - Y_Z \tag{2.93}$$

with the heats of formation given by

$$H_{f_1} = H_{f_Z} \qquad H_{f_2} = 0. \tag{2.94}$$

Since each of the constituents is assumed to be a perfect gas with $\gamma = 5/3$ for Z and $\gamma = 4/3$ for Z_2, the constant volume specific heats are

$$\begin{aligned} C_{v_1} &= \frac{R}{(\gamma_Z - 1)\hat{m}_Z} = \frac{3R}{2\hat{m}_Z} = 3R_{Z_2} \\ C_{v_2} &= \frac{R}{(\gamma_{Z_2} - 1)\hat{m}_{Z_2}} = \frac{3R}{\hat{m}_{Z_2}} = 3R_{Z_2} = C_{v_1} \end{aligned} \tag{2.95}$$

where $R_{Z_2} = \mathcal{R}/\hat{m}_{Z_2}$ is the gas constant for the molecule. The constant pressure specific heats are given by

$$\begin{aligned} C_{p_1} &= C_{v_1} + \tfrac{\mathcal{R}}{\hat{m}_Z} = 5R_{Z_2} \\ C_{p_2} &= C_{v_2} + \tfrac{\mathcal{R}}{\hat{m}_{Z_2}} = 4R_{Z_2}. \end{aligned} \tag{2.96}$$

The thermal equation of state is given by

$$p = \rho \mathcal{R} T \left[\frac{Y_1}{\hat{m}_Z} + \frac{1 - Y_1}{\hat{m}_{Z_2}} \right] = \rho R_{Z_2} T (1 + Y_1). \tag{2.97}$$

The caloric equation of state for ideal dissociating gas is [78,136]

$$\frac{\epsilon}{\rho} = R_{Z_2} (3T + Y_1 \theta_d) + \frac{u^2 + v^2}{2} \tag{2.98}$$

where θ_d is the characteristic temperature for dissociation (59,500 K for O_2). The corresponding multicomponent Equation (2.33) yields

$$\frac{\epsilon}{\rho} = 3R_{Z_2}(T - T_0) + Y_1 H_{f_1} - R_{Z_2} T_0 (1 + Y_1) + \frac{u^2 + v^2}{2}. \tag{2.99}$$

Comparing the previous two equations yields the expected

$$H_{f_1} = R_{Z_2} \theta_d \qquad \text{and} \qquad T_0 = 0. \tag{2.100}$$

The nonequilibrium chemical source term, in dimensional form, is given by [52]

$$\dot{W}_1 = \frac{C_f}{\hat{m}_Z} T^\eta \rho^2 \left[(1 - Y_1) e^{-\theta_d / T} - \frac{\rho}{\rho_d} Y_1^2 \right] \tag{2.101}$$

where C_f is a constant which depends upon the collision cross-section between molecules and those between atoms and molecules. The constant ρ_d is the characteristic density for dissociation (1.5×10^5 kg/m^3 for O_2). Similarly Equation (2.25) yields the following dimensional form

$$\dot{W}_1 = \rho \left[K_f (1 - Y_1) - 2K_b \frac{\rho Y_1^2}{\hat{m}_Z} \right]. \tag{2.102}$$

Choosing

$$K_f = A_f T^\eta e^{-\theta_d / T} \qquad \text{and} \qquad K_b = A_b T^\eta \tag{2.103}$$

yields dimensional form

$$\dot{W}_1 = \rho A_f T^\eta \left[(1 - Y_1) e^{-\theta_d / T} - \frac{2\rho}{\hat{m}_Z} \frac{A_b}{A_f} Y_1^2 \right]. \tag{2.104}$$

Comparing Equations (2.101) and (2.104) yields

$$A_f = \frac{C_f}{\hat{m}_Z}\rho \qquad \text{and} \qquad A_b = \frac{\hat{m}_Z A_f}{2\rho_d}. \tag{2.105}$$

The non-dimensional form of the source term (Eq. 2.101) with Equations (2.51) and (2.53) is

$$\dot{W}_1 = \Phi T^\eta \rho^2 \left[(1-Y_1)e^{-\theta_d/T} - \frac{\rho}{\rho_d}Y_1^2\right] \tag{2.106}$$

where the non-dimensional reaction parameter is given by

$$\Phi = \frac{C_f T_r^\eta \rho_r L_r}{\hat{m}_Z \sqrt{p_r/\rho_r}} \tag{2.107}$$

and $\dot{W}_1$, ρ, T, ρ_d and θ_d are non-dimensional variables in Equation (2.106). The pre-exponential factor A_f in terms of Φ becomes

$$A_f = \Phi\rho\frac{u_r}{T_r^\eta L_r}. \tag{2.108}$$

Here again ρ is the non-dimensional density. The rate parameter Φ varies from zero for frozen flow, to infinity for equilibrium flow.

2.8.2 Hydrogen-Air Combustion Model

For a scramjet combustor Rogers and Chinitz [115] used a 28 reaction H-O mechanism to propose a two reaction model for combustion of hydrogen in air. Nitrogen was regarded as inert. The model is applicable for temperatures between 1000 to 2000 K and for equivalence ratios between 0.2 and 2.0. The model consists of the following two steps

$$\begin{aligned} H_2 + O_2 &\rightleftharpoons 2OH \\ H_2 + 2OH &\rightleftharpoons 2H_2O. \end{aligned} \tag{2.109}$$

The first controls the reaction of the fuel and oxidizer species through the ignition delay period, whereas the second step predominates during the combustion phase when the major heat release and product formation occurs. The model adequately represents the physics of hydrogen combustion in air but produces an extremely large disparity in the time-scales associated with the two reactions. Hence this model can be used for testing the robustness of a numerical scheme in overcoming the resulting stiffness.

The forward rate coefficients for the reaction are determined to be functions of temperature-and equivalence ratio ϕ with

$$\begin{aligned} A_{f_1} &= (31.433/\phi \ +8.917\phi \ -28.95 \ ) \times 10^{44} \quad m^3/(kmole.s) \\ A_{f_2} &= (1.333/\phi \ -0.833\phi \ +2.00 \ ) \times 10^{58} \quad m^6/(kmole^2.s) \\ \eta_{f_1} &= -10 \qquad\qquad \eta_{f_2} = -13 \\ E_{f_1}/\mathcal{R} &= 2448.4 \ K \qquad E_{f_2}/\mathcal{R} = 18940.6 \ K \end{aligned} \tag{2.110}$$

here the equivalence ratio ϕ is defined as the fuel to air ratio divided by the stoichiometric fuel to air ratio, thus for the following *complete* reaction

$$2H_2 + (O_2 + 3.76N_2) \quad \longrightarrow \quad 2H_2O + 3.76N_2 \tag{2.111}$$

the fuel to air ratio becomes

$$\frac{f}{a} = \frac{2\hat{m}_H}{\hat{m}_O + 3.76\hat{m}_N}\phi = 0.02937\phi. \tag{2.112}$$

The mass fraction of hydrogen is then given by

$$Y_{H_2} = \frac{\phi}{\phi + 34.048}. \tag{2.113}$$

The backward rate coefficients are determined from the law of mass action with the following equilibrium constants [42]

$$\begin{aligned} K_{c_1} &= 26.164e^{-8992/T} \\ K_{c_2} &= 2.682 \times 10^{-9} T e^{+69415/T} \ \ m^3/kmol. \end{aligned} \tag{2.114}$$

The chemical source terms are given by Equations (2.25). Since this chemistry model is not valid below 1000 K an ignition temperature must be specified. This temperature for hydrogen-air combustion is itself about 1000 K. For temperatures below the ignition temperature the chemical source terms are set equal to zero.

For premixed flows 7 equations (4 fluid and 3 species) define the flow. This is because Y_{N_2} is constant and $\sum Y_s = 1$. However, when the fuel is injected Y_{N_2} is only piecewise constant and hence 8 equations need to be solved.

The constant pressure specific heat for each species has been computed from non-linear thermodynamic equations in Reference [134] and a least square regression is performed for temperatures between 300 and 2500 K. These approximations are

$$\begin{aligned}
C_p(O_2) &= 30.559 + 3.4485 \times 10^{-3}\ T \qquad kJ/kmol K \\
C_p(OH) &= 28.071 + 3.0943 \times 10^{-3}\ T \qquad kJ/kmol K \\
C_p(H_2) &= 27.290 + 3.3530 \times 10^{-3}\ T \qquad kJ/kmol K \\
C_p(H_2O) &= 32.469 + 8.6358 \times 10^{-3}\ T \qquad kJ/kmol K \\
C_p(N_2) &= 29.282 + 3.0233 \times 10^{-3}\ T \qquad kJ/kmol K.
\end{aligned} \tag{2.115}$$

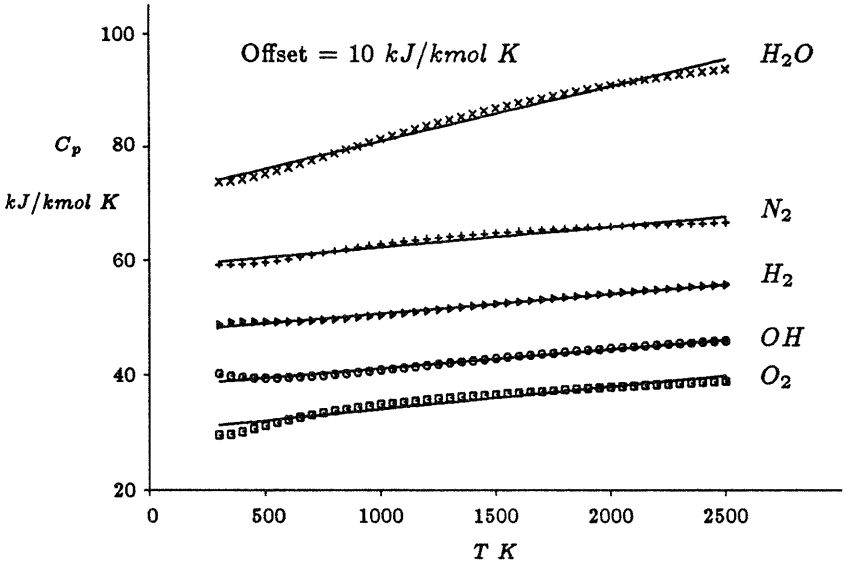

Figure 2.1: Variation of constant pressure specific heat with temperature.

Figure (2.1) shows the variation of constant pressure specific heat with temperature, the symbols represent the data from Reference [134]. Also shown are the linear profiles which fit the data reasonably well. The vertical scale corresponds to the oxygen curve and the rest of the curves are displaced by the indicated offset.

Chapter 3

Integration Scheme

The integration basis for the present algorithm is a generalization of the second order Lax-Wendroff, finite volume, cell-vertex scheme originally published by Ni [96]. Another cell-vertex scheme which is very similar to the Ni scheme is due to Hall [60,61]. The generalization introduces chemical source terms and spatio-temporal adaptation. The state variables U, the source terms W, *etc.* are stored at the nodes and each cell is integrated independently based upon the nodal values of these vectors. Ni made use of a multiple-grid accelerator for his steady state interest but that is inappropriate for unsteady situations discussed in this thesis.

Section (3.1) deals with the integral form of the governing equations. The integration procedures for both one and two-dimensional cases are developed in Sections (3.2) and (3.4). A discussion of artificial viscosity modelling for one spatial dimension is contained in Section (3.3) and this is extended to cover two spatial dimensions in Section (3.6). The treatment of 2-D spatial interfaces is discussed in Section (3.5).

3.1 Integral Form of Governing Equations

To integrate the mathematical model numerically the governing equations must be discretized in both space and time. Instead of immediately discretizing the Euler equations the governing equations are often cast first into integral form and then the flux is balanced across computational units which are known as the cells. This approach is referred to as the finite volume or cell method. Time is divided into finite intervals called time-steps. The approximate numerical scheme is advanced through each time-

step for all computational cells. With smaller cell dimensions and shorter time-steps, the numerical solution is believed to approach the exact solution of the original partial differential equations for a given choice of boundary and initial conditions. A finite volume calculation on the cells ensures conservation of global and species mass, momenta and energy on the smallest computational units and thereby leads to conservation of these quantities globally over both the space and time dimensions. The finite volume approach also allows one to deal with complicated geometries without the complexity of curvilinear coordinates [107]. Thus the basic cell units can be triangles, quadrilaterals or a combination of other higher dimensional polygons. Only the coordinates of the nodes of the cells are really necessary and non-orthogonal curvilinear coordinates can be employed to define the set of volumes.

The governing equations (2.47) are well suited for finite volume discretization with the integral form since they have been formulated in conservation law form. The integral form of the governing equations can be expressed as

$$\int_\Omega \frac{\partial U}{\partial t} dV + \int_\Omega \left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} \right) dV \;=\; \int_\Omega W dV. \tag{3.1}$$

Here Ω is the region of validity of the equations and $\partial\Omega$ is the boundary surface of this fixed region. Using the divergence theorem, the integral of flux vectors can be transformed into a surface integral along the cell boundaries

$$\frac{\partial}{\partial t} \int_\Omega U dV + \oint_{\partial\Omega}^{CCW} (F, G) \cdot \hat{n} dA \;=\; \int_\Omega W dV \tag{3.2}$$

where $\hat{n}$ is a unit normal pointing outward from the surface $\partial\Omega$. The superscript on the surface integral accents the counter-clock-wise orientation. For the Cartesian frame of reference in two spatial dimensions the unit normal vector can be decomposed as

$$\hat{n} \;=\; \frac{dy}{ds}\hat{i} - \frac{dx}{ds}\hat{j} \tag{3.3}$$

thereby yielding

$$A_\Omega \frac{dU_\Omega}{dt} + \oint_{\partial\Omega}^{CCW} (F dy - G dx) \;=\; A_\Omega W_\Omega. \tag{3.4}$$

Here U_Ω and W_Ω are taken to be cell averaged values; for example

$$U_\Omega \;=\; \frac{\int_\Omega U dA}{\int_\Omega dA} \;=\; \frac{1}{A_\Omega} \int_\Omega U dA \tag{3.5}$$

Thus the changes occurring in time Δt_C for some cell C are given by

$$\frac{\Delta U_C}{\Delta t_C} = \left.\frac{dU}{dt_C}\right|_C = W_C + \frac{1}{A_\Omega}\int_{\partial C}^{CW} (F dy - G dx). \tag{3.6}$$

The quantity ΔU_C will be referred to as the *first order cell change in time* or simply as *cell change*. The process of calculating cell change is usually termed as flux balancing and is principally the summing of the quantities $(F\Delta y - G\Delta x)$ over the cell faces and the source terms over the cell volume. The corresponding equation for first order changes in the computational coordinates (ξ, η) is

$$\frac{\Delta \tilde{U}_C}{\Delta t_C} = \tilde{W}_C + \frac{1}{\Delta\xi\Delta\eta}\int_{\partial C}^{CW} \left(\tilde{F} d\eta - \tilde{G} d\xi\right). \tag{3.7}$$

It will be proved later in Section (3.4) that the cell change is the same whether computed from Equation (3.6), or (3.7) and then transformed back to physical coordinates. The scheme developed in this chapter will be referred to as the Ni scheme, although the original Ni algorithm [96] involves neither chemistry nor spatio-temporal adaptation. The discretized version of the overall Ni scheme is obtained by coupling the cell changes with the residuals at the nodes. This will now be discussed for both one and two dimensional spatial systems.

3.2 Integration Scheme for One Spatial Dimension

The development of the integration scheme in one spatial dimension is important in understanding the concept of time-strides and artificial viscosity and for the studies pertaining to stability analysis. Consider the cells B and C adjacent to node j in Figure (3.1) with a constant time-step Δt for both cells.

The temporal change in state at node j is

$$\delta U_j \equiv U_j^{n+1} - U_j^n = \left.\frac{\partial U}{\partial t}\right|_j^n \Delta t + \frac{1}{2}\left.\frac{\partial^2 U}{\partial t^2}\right|_j^n \Delta t^2 + O(\Delta t^3) \tag{3.8}$$

or, using Equations (2.45),

$$\delta U_j = \left(W - \frac{\partial F}{\partial x}\right)\Delta t + \frac{\Delta t^2}{2}\left[qW_U\left(W - \frac{\partial F}{\partial x}\right) - \frac{\partial}{\partial x}\left\{F_U\left(W - \frac{\partial F}{\partial x}\right)\right\}\right]. \tag{3.9}$$

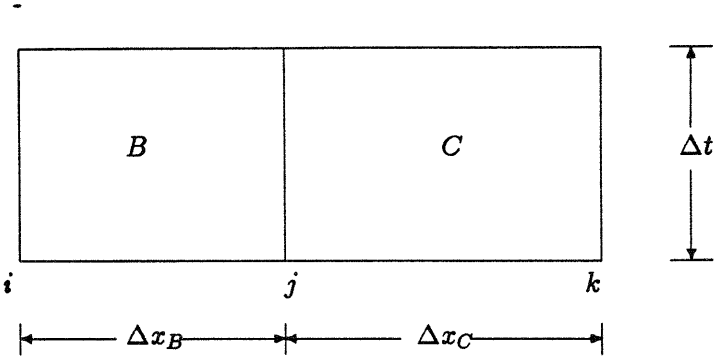

Figure 3.1: Finite volumes adjacent to node j.

The subscript j and superscript n have been omitted for simplicity. The factor q appears in this equation so that assigned values of 0 or 1 will exclude or include a second order source term. The remaining second order flux terms are essential for stability of Euler equations and hence are always retained. The significance will be clearer when the stability analysis of a model problem is discussed in the next chapter. The first order source term (Eq. 3.11) is always included (whether explicit or implicit) irrespective of the inclusion of a second order source term. The Jacobians in the above equation are defined, for example, as

$$W_U = \left(\frac{\partial W}{\partial U}\right)_j^n. \tag{3.10}$$

The flux balance for cell C, for example, yields the *cell change*

$$\Delta U_C = W_C \Delta t_C + (F_j - F_k)\frac{\Delta t_C}{\Delta x_C} \tag{3.11}$$

in which W_C may be modelled as an average for the cell, *i.e.*,

$$W_C = (W_j + W_k)/2. \tag{3.12}$$

Alternatively, for a more accurate contribution to node j use can be made of a ΔU_{jC} based on choosing the source term as W_j, in which case the cell change varies with the nodal source terms, *viz.*,

$$\Delta U_{jC} = W_j \Delta t_C + (F_j - F_k)\frac{\Delta t_C}{\Delta x_C}. \tag{3.13}$$

This represents an accurate contribution to node j because the source terms and Jacobians in Equations (3.8) and (3.9) are based upon nodal values rather than the cell values. Since accuracy is not imperative to a determination of cell time-steps, Equation (3.11) will be used as a basis for determination of temporal resolution Δt, whereas Equation (3.13) will be actually used for determining the residuals at the nodes. The criterion for temporal resolution is developed and explained in Chapter 5.

In terms of a non-uniformity grid parameter ϵ_j at node j

$$\epsilon_j = \frac{\Delta x_B - \Delta x_C}{\Delta x_B + \Delta x_C} \tag{3.14}$$

a second order accurate Taylor series expression for the rate of change of a scalar variable ϕ can be defined as

$$\left.\frac{\partial \phi}{\partial x}\right|_j = \frac{1+\epsilon_j}{2\Delta x_C}(\phi_k - \phi_j) - \frac{1-\epsilon_j}{2\Delta x_B}(\phi_i - \phi_j) + O\left(\Delta x_B \Delta x_C\right). \tag{3.15}$$

Note that for uniform grids $\epsilon_j \equiv 0$ and for embedding involving uniform base grids ϵ_j will be either $\frac{1}{3}$ when $\Delta x_B = 2\Delta x_C$ or $-\frac{1}{3}$ when $\Delta x_C = 2\Delta x_B$ at the extreme edges of the embedded regions. Hence the *spatial interfaces* for one-dimensional spatial grids can be defined to be those nodes which are at boundary of disparate cell sizes with $|\epsilon_j| \geq \frac{1}{3}$. Using the above expression and Equation (3.13) the following terms in Equation (3.9) can be evaluated; *i.e.*, the first order node change is

$$\begin{aligned}\left(W - \frac{\partial F}{\partial x}\right)_j^n \Delta t &= \frac{1-\epsilon_j}{2}\left[W_j \Delta t + \frac{\Delta t}{\Delta x_B}(F_i - F_j)\right] + \frac{1+\epsilon_j}{2}\left[W_j \Delta t + \frac{\Delta t}{\Delta x_C}(F_j - F_k)\right] \\ &= \frac{1-\epsilon_j}{2}\Delta U_{jB} + \frac{1+\epsilon_j}{2}\Delta U_{jC}\end{aligned} \tag{3.16}$$

and the second order *source change* is

$$\begin{aligned}\frac{\Delta t^2}{2} W_U \left(W - \frac{\partial F}{\partial x}\right)_j^n &= \frac{\Delta t}{2} W_{U_j}\left(\frac{1-\epsilon_j}{2}\Delta U_{jB} + \frac{1+\epsilon_j}{2}\Delta U_{jC}\right) \\ &= \frac{\Delta t}{2}\left(\frac{1-\epsilon_j}{2}\Delta W_{jB} + \frac{1+\epsilon_j}{2}\Delta W_{jC}\right).\end{aligned} \tag{3.17}$$

The definitions of ΔW_{jB} and ΔW_{jC} are similar to the forms in Equations (3.20) shown below. The second order flux change is now

$$\begin{aligned}-\frac{\Delta t^2}{2}\frac{\partial}{\partial x}\left\{F_U\left(W - \frac{\partial F}{\partial x}\right)\right\} &= -\frac{\Delta t^2}{2}\left\{\frac{1+\epsilon_j}{\Delta x_C}\left[F_{U_C}\left(W - \frac{\partial F}{\partial x}\right)_{j+\frac{k}{2}} - F_{U_j}\left(W - \frac{\partial F}{\partial x}\right)_j\right]\right. \\ &\quad \left. -\frac{1-\epsilon_j}{\Delta x_B}\left[F_{U_B}\left(W - \frac{\partial F}{\partial x}\right)_{j+\frac{j}{2}} - F_{U_j}\left(W - \frac{\partial F}{\partial x}\right)_j\right]\right\}.\end{aligned} \tag{3.18}$$

In this expression $\frac{i+k}{2}$ denotes the mid-value in between the nodes j and k. After some algebra this can be discretized to

$$-\frac{\Delta t^2}{2}\frac{\partial}{\partial x}\left\{F_U\left(W-\frac{\partial F}{\partial x}\right)\right\} = \frac{\Delta t}{\Delta x_B}\left[\frac{1-\epsilon_j}{2}\Delta F_B+\epsilon_j\Delta F_{jB}\right]+ \frac{\Delta t}{\Delta x_C}\left[-\frac{1+\epsilon_j}{2}\Delta F_C+\epsilon_j\Delta F_{jC}\right]. \quad (3.19)$$

Here the various *Jacobian changes* are defined as, for example,

$$\Delta F_C = \left.\frac{\partial F}{\partial U}\right|_C \Delta U_C \ , \qquad \Delta F_{jC} = \left.\frac{\partial F}{\partial U}\right|_j \Delta U_{jC}. \quad (3.20)$$

When the three terms contributing to δU_j (Eqs. 3.16,3.17,3.19) are added the resulting overall change can be decomposed into distinct contributions from cells B and C, *i.e.*,

$$\delta U_j = \delta U_{jB} + \delta U_{jC} \quad (3.21)$$

where

$$\begin{aligned}\delta U_{jB} &= \tfrac{1-\epsilon_j}{2}\left[\Delta U_{jB}+\tfrac{\Delta t_B}{\Delta x_B}\left(\Delta F_B+\tfrac{2\epsilon_j}{1-\epsilon_j}\Delta F_{jB}\right)+q\tfrac{\Delta t_B}{2}\Delta W_{jB}\right]\\ \delta U_{jC} &= \tfrac{1+\epsilon_j}{2}\left[\Delta U_{jC}-\tfrac{\Delta t_C}{\Delta x_C}\left(\Delta F_C-\tfrac{2\epsilon_j}{1+\epsilon_j}\Delta F_{jC}\right)+q\tfrac{\Delta t_C}{2}\Delta W_{jC}\right]\end{aligned} \quad (3.22)$$

are the ***distribution formulae***. For frozen flows on uniform grids ($\epsilon_j = 0$, $W = 0$) these expressions reduce to those in Ni's paper [96]. Also note that the time-step Δt is now replaced by Δt_B and Δt_C for cells B and C respectively. Hence the distribution formulae can now be used to update the cells adjoining a common node with different time-steps. A node adjoining cells with different time-steps will be referred to as ***nodit*** which is an acronym for "Node Of DIfferent Time-steps". It can also be noted that if the integration is carried out on a cell by cell basis then the contributions to the nodes of a given cell only involve information based on nodes of that cell, *i.e.*, the contributions do not involve information from the nodes of the neighboring cells. This property is extremely beneficial when adaptive grid structures are considered. The distribution formulae in the above form do not involve artificial viscosity and its inclusion is discussed next.

3.3 Artificial Viscosity in One Spatial Dimension

An explicit artificial viscosity is needed for the following reasons:

- to suppress odd-even decoupling modes associated with the integration scheme
- to stabilize captured shocks in transonic and supersonic regimes.

One must exercise care to ensure that the numerical smoothing does not contaminate the solution above some acceptable level. This issue becomes even more important when real viscous and diffusion terms are involved.

The explicit artificial viscosity for the original Ni scheme [96] for a uniform grid is of the form

$$\delta U_j^* = \frac{\sigma}{4}\Delta t \Delta x \left.\frac{\partial^2 U}{\partial x^2}\right|_j^n. \tag{3.23}$$

This ***viscous change*** is added in a discretized form to the distribution formulae (Eq. 3.22) and implies the following modified differential equation

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} = W + \frac{\sigma}{4}\Delta x \frac{\partial^2 U}{\partial x^2}. \tag{3.24}$$

The artificial viscosity coefficient σ was regarded as constant in Ni's paper, who had not considered high supersonic flows. For flows involving strong shocks a relatively large value of σ is needed in their vicinity. A constant value of σ would result in excessive errors due to artificial viscosity in smooth regions of the flow field. Hence it is desirable to use formulations in which the artificial viscosity coefficient will be small enough in smooth regions to suppress spurious oscillations and large enough in the vicinity of strong shocks for adequate shock capturing. Another desirable property for the artificial viscosity would be a non-convective conservative formulation. Hence the viscous change should be of the form

$$\delta U_j^* = \Delta t \frac{\partial^2}{\partial x^2}\left(\frac{\Delta x^* \sigma U}{4}\right)_j. \tag{3.25}$$

Since it is not yet clear which Δx to use for non-uniform grids at node j, the symbol Δx^* is used tentatively. A Taylor series expansion for a second derivative of a scalar function ϕ is similar to Equation (3.15) and has the form

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_j = \frac{1-\epsilon_j}{\Delta x_C^2}(\phi_k - \phi_j) + \frac{1+\epsilon_j}{\Delta x_B^2}(\phi_i - \phi_j) + \epsilon_j O(\Delta x_B + \Delta x_C) + O(\Delta x_B \Delta x_C). \tag{3.26}$$

Note that unlike Equation (3.15) this expression is first order accurate if $\epsilon_j \neq 0$ and becomes second order accurate for uniform grids. Using this equation with Equation (3.25) gives

$$\delta U_j^* = \frac{\Delta t}{4}\frac{(1-\epsilon_j)}{\Delta x_C^2}\Delta x_C^* (\sigma_k U_k - \sigma_j U_j) + \frac{\Delta t}{4}\frac{(1+\epsilon_j)}{\Delta x_B^2}\Delta x_B^* (\sigma_i U_i - \sigma_j U_j). \quad (3.27)$$

A logical choice for Δx^* would be

$$\Delta x_B^* = \Delta x_B \qquad \text{and} \qquad \Delta x_C^* = \Delta x_C \quad (3.28)$$

which yields

$$\delta U_j^* = -\frac{\Delta t}{\Delta x_C}\frac{(1-\epsilon_j)}{4}(\sigma_j U_j - \sigma_k U_k) + \frac{\Delta t}{\Delta x_B}\frac{(1+\epsilon_j)}{4}(\sigma_i U_i - \sigma_j U_j). \quad (3.29)$$

Note that in this equation the term $(1+\epsilon_j)$ appears with the quantities corresponding to cell B unlike the rest of the terms in the derivation of δU_j in Equation (3.22). Hence in order to make the coefficients of the terms consistent the following choice is made

$$\Delta x_B^* = \Delta x_C \qquad \text{and} \qquad \Delta x_C^* = \Delta x_B. \quad (3.30)$$

Hence

$$\delta U_j^* = -\frac{\Delta t}{\Delta x_C}\frac{1+\epsilon_j}{2}\Psi_C + \frac{\Delta t}{\Delta x_B}\frac{1-\epsilon_j}{2}\Psi_B \quad (3.31)$$

where, for example,

$$\Psi_C = \frac{\sigma_j U_j - \sigma_k U_k}{2}. \quad (3.32)$$

For uniform meshes ($\epsilon_j = 0$) with $i = j-1$ and $k = j+1$, the artificial viscosity contribution at node j is

$$\delta U_j^* = \frac{\Delta t}{4\Delta x}(\sigma_{j-1}U_{j-1} - 2\sigma_j U_j + \sigma_{j+1}U_{j+1}). \quad (3.33)$$

Hence the sum of all the viscous changes for all the interior nodes satisfies

$$\sum_{j=2}^{J-1}\delta U_j^* = \frac{\Delta t}{4\Delta x}(\sigma_1 U_1 - \sigma_2 U_2 - \sigma_{J-1}U_{J-1} + \sigma_J U_J) \quad (3.34)$$

but the contribution from the first cell at node 1 is $-\frac{\Delta t}{4\Delta x}(\sigma_1 U_1 - \sigma_2 U_2)$ whereas the contribution from the last cell at node J is $\frac{\Delta t}{4\Delta x}(\sigma_{J-1}U_{J-1} - \sigma_J U_J)$ as given by Equation (3.31). Hence the artificial viscosity contribution is conservative and at the same

time non-convective (*i.e.*, there are no terms of the form $\partial(\sigma U)/\partial x$ in Eq. 3.31) on uniform grids. The overall distribution formulae for cell C can now be written as

$$\begin{aligned} \delta U_{jC} &= \tfrac{1+\epsilon_j}{2}\left[\Delta U_{jC} - \tfrac{\Delta t_C}{\Delta x_C}\left(\Delta F_C - \tfrac{2\epsilon_j}{1+\epsilon_j}\Delta F_{jC} + \Psi_C\right) + q\tfrac{\Delta t_C}{2}\Delta W_{jC}\right] \\ \delta U_{kC} &= \tfrac{1-\epsilon_k}{2}\left[\Delta U_{kC} + \tfrac{\Delta t_C}{\Delta x_C}\left(\Delta F_C + \tfrac{2\epsilon_k}{1-\epsilon_k}\Delta F_{kC} + \Psi_C\right) + q\tfrac{\Delta t_C}{2}\Delta W_{kC}\right]. \end{aligned} \tag{3.35}$$

The second difference of pressure is commonly used to scale the artificial viscosity coefficient [68]. This is because the second differences are considerably larger (order unity) for regions in the vicinity of shocks compared to those in smooth regions (order Δx^2 for Ni scheme). Since pressure is constant across contact surfaces, density is used in the present work for scaling artificial viscosity. Furthermore normalized first differences are used in this study instead of second differences. Consider the second difference at node j

$$\left|\frac{\partial^2 \rho}{\partial x^2}\Delta x^2\right| = |(\rho_{j-1} - \rho_j) + (\rho_{j+1} - \rho_j)| \leq |\rho_{j-1} - \rho_j| + |\rho_{j+1} - \rho_j|. \tag{3.36}$$

Hence the sum of the two first differences for the cells adjoining the nodes j is even greater than the magnitude of second difference at this node. The first differences will be of order unity in the vicinity of strong shocks and would be of second order for the current scheme in the smooth regions. This is not the first time that first differences of density have been used for scaling the artificial viscosity coefficient; specifically Hall and Salas [61] have used a different form of first differences. Defining the normalized scaling for cell C as

$$\kappa_C = \left|\frac{\rho_j - \rho_k}{\rho_j + \rho_k}\right| \tag{3.37}$$

the nodal artificial viscosity coefficient can be assigned as

$$\sigma_j = \sigma_{min} + \frac{\delta}{2}(\kappa_B + \kappa_C). \tag{3.38}$$

Here σ_{min} is the minimum amount of artificial viscosity which shall be deemed necessary to suppress odd-even decoupling in the smooth regions and δ is a constant which is chosen so that $\sigma_j \in [\sigma_{min}, \sigma_{max}]$ with σ_{max} being the maximum user supplied viscosity. The artificial viscosity can be kept within bounds by the following formula

$$\delta = \frac{\sigma_{max} - \sigma_{min}}{\max\{\kappa_C\}} \tag{3.39}$$

where $\max\{\kappa_C\}$ is the maximum value of the normalized scaling for all the cells in the domain. Typical values for σ fall within 0.01 and 0.2 for most one dimensional results shown in Chapter 8.

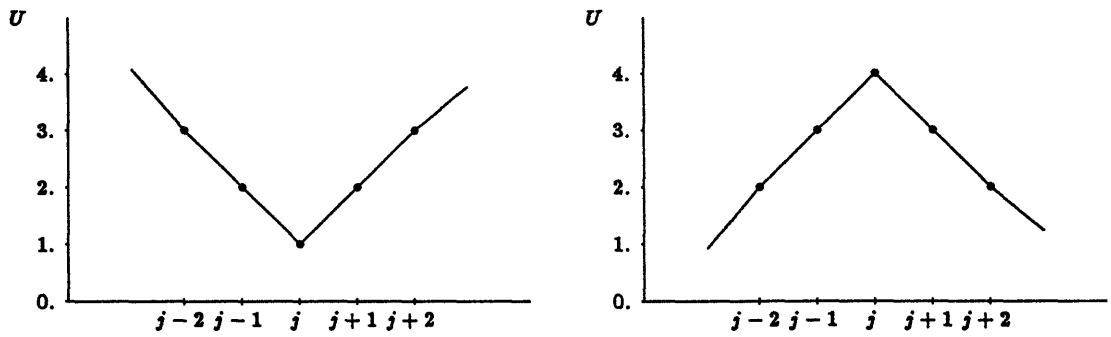

Figure 3.2: Distribution before the application of artificial viscosity.

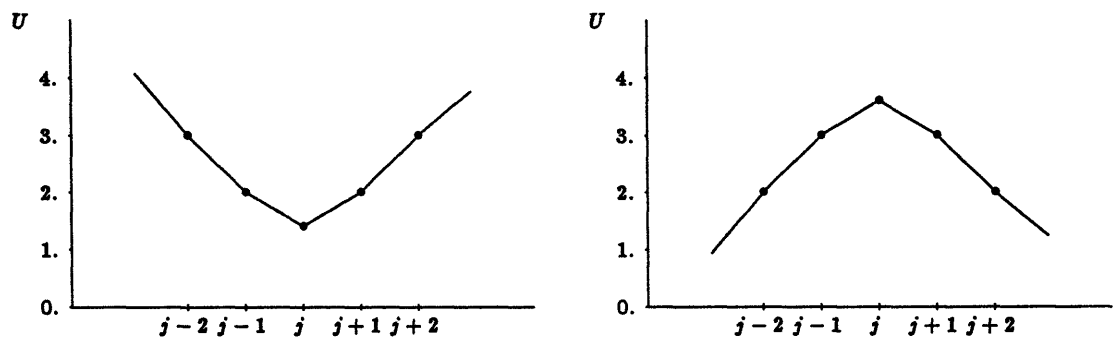

Figure 3.3: Distribution after the application of artificial viscosity.

In order to understand how the artificial viscosity suppresses spurious oscillations consider the two situations as shown in Figure (3.2) before the application of artificial viscosity. These correspond to a spurious valley and peak for one of the components of the state vector. Further suppose that the artificial viscosity coefficient is constant and

the value of $\sigma\Delta t/4\Delta x = 0.2$. For simplicity the slopes of the distribution of U on the two sides of node j are regarded as constants. As evident from Equation (3.33) the artificial viscosity contribution at nodes $j-1$ and $j+1$ is identically zero. For the downward pointing spike the artificial viscosity contribution at node j is $+0.4$, whereas that for the upward pointing spike is -0.4. Hence the amplitude of the spikes decreases after the application of artificial viscosity as indicated by Figure (3.3). Thus the numerical diffusion has the same form and effect as physical diffusion and reduces the amplitudes of the solution harmonics without altering their phases.

3.4 Integration Scheme for Two Spatial Dimensions

The changes in the state vector for the cell centers (Eq. 3.6) must also be related to the temporal variation at the nodes for the 2-D case. Consider cells A through D in the computational domain and adjacent to node i in Figure (3.4). Since the generalized coordinate transformation $\xi = \xi(x,y)$, $\eta = \eta(x,y)$ is arbitrary, it can be used to map each physical cell onto equi-dimensional rectangles for convenience while the physical grid conforms to the boundary shapes. The computational grid is locally 1-1 and onto for each cell and may not be so for the entire domain when the cells are subdivided and *spatial interfaces* are created. Hence for a local uniform grid in the computational coordinates (ξ,η) with constant $\Delta\xi$ and $\Delta\eta$ for the cells A through D and with constant time-steps, the temporal change in state at node i node is given by the Taylor series expansion

$$\delta\tilde{U}_i \equiv \tilde{U}_i^{n+1} - \tilde{U}_i^n = \left.\frac{\partial\tilde{U}}{\partial t}\right|_i^n \Delta t + \frac{1}{2}\left.\frac{\partial^2\tilde{U}}{\partial t^2}\right|_i^n \Delta t^2 + O(\Delta t^3). \tag{3.40}$$

The variations in cell time-steps will be allowed once the *distribution formulae* (Eq. 3.70) are derived [96]. Using Equation (2.47), the first order term or FOCIT (First Order Change In Time) in Equation (3.40) can be written as

$$\text{FOCIT} = \left.\frac{\partial\tilde{U}}{\partial t}\right|_i^n \Delta t = \left(\tilde{W}^* - \tilde{F}_\xi - \tilde{G}_\eta\right)\Delta t \tag{3.41}$$

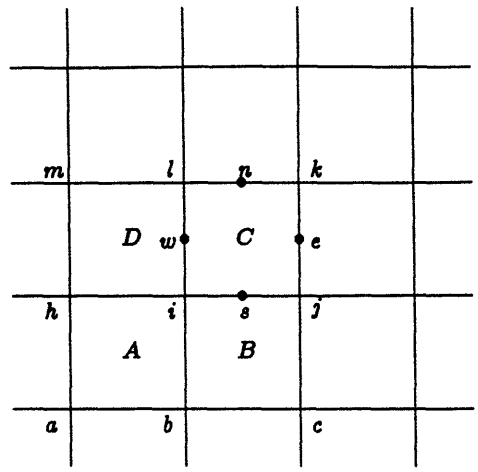

Figure 3.4: Computational grid for flux balance.

where the asterisk on W indicates that the source term can be treated either explicitly or implicitly. The implicit source vector is useful when the chemical reactions would otherwise impose a severe time-step restriction due to the stability considerations involving chemical time scales, and would thereby make time-steps minuscule compared to resolution requirements. However, it is essential to realize that such implicit modelling is desirable only when the stability dictated time-step is small compared to the resolution requirement. The latter will be discussed with considerations which arise for temporal adaptation. Although implicit modelling may be advantageous in overcoming the reaction stability limitations, this approach should not be applied to avoid local rapid chemical adjustments. Of course when interest is limited to the steady state the implicit advantage can be fully utilized in by-passing the resolution requirements [24,42,114,122], but only if the real gas behavior is independent of transient history, which is not always clear.

The use of only the first order term in the Taylor series expansion yields an unconditionally unstable scheme. However, the scheme can be stabilized by considering the next term in the Taylor series expansion and this process is frequently termed as Lax-

Wendroff time-stepping [76]. The inclusion of an additional term results in an inherent upwind biasing which admits correct wave propagation phenomenon. The second order change in time contribution or SOCIT in Equation (3.40) is again determined by appropriately differentiating the original differential equations, *i.e.*,

$$\begin{aligned} \text{SOCIT} \;=\; \tfrac{1}{2}\tfrac{\partial^2 \tilde{U}}{\partial t^2}\Big|_i^n \Delta t^2 \;&=\; \tfrac{\Delta t^2}{2}\tfrac{\partial}{\partial t}\left(\tilde{W} - \tfrac{\partial \tilde{F}}{\partial \xi} - \tfrac{\partial \tilde{G}}{\partial \eta}\right) \\ &=\; q\tfrac{\Delta t^2}{2}\tilde{W}_U\tilde{U}_t - \tfrac{\Delta t^2}{2}\tfrac{\partial}{\partial \xi}\left(\tilde{F}_U\tilde{U}_t\right) - \tfrac{\Delta t^2}{2}\tfrac{\partial}{\partial \eta}\left(\tilde{G}_U\tilde{U}_t\right). \end{aligned} \tag{3.42}$$

As pointed out earlier the factor q is assigned values 1 or 0 to include or exclude the second order source term. The Jacobians are defined, for example, as

$$\tilde{W}_U \;=\; \left(\frac{\partial \tilde{W}}{\partial \tilde{U}}\right)_i^n. \tag{3.43}$$

These nodal Jacobians subsequently will be replaced by their cellular representation for the system of two spatial dimensions.

The cells in a physical domain are depicted in Figure (3.5). Additional divided cells bordering cell C are shown in this figure. It is reasserted that the transformation with constant $\Delta\xi$ and $\Delta\eta$ is applied only for cells adjoining the *usual* nodes, and not cells with a node at the mid-point of a spatial interface (such as nodes e and n in the physical grid). These spatial interfaces are one or more faces of a given undivided cell if one or more cells adjacent to it are divided. The treatment for the latter nodes will be discussed separately. It will be proven first that the cell change can be obtained either from a cell in the physical grid or from a corresponding cell in the computational grid (with $\Delta\tilde{U}_C = \Delta U_C/J$), provided that the metrics are specified in a certain manner. For the sake of this proof, average values of the corner node fluxes will be used for the respective sides of a cell and the middle node values will not be accounted. For example, the west and north face F-fluxes for this proof are

$$F_W \;=\; \frac{F_l + F_i}{2}, \qquad F_N \;=\; \frac{F_k + F_l}{2}.$$

The proof for other variations of fluxes involving the middle nodes of the faces can be verified in a similar manner. For example, the north face F-flux for the cell C in Figure (3.5) could be defined as

$$F_N \;=\; \frac{F_l + 2F_n + F_k}{4}.$$

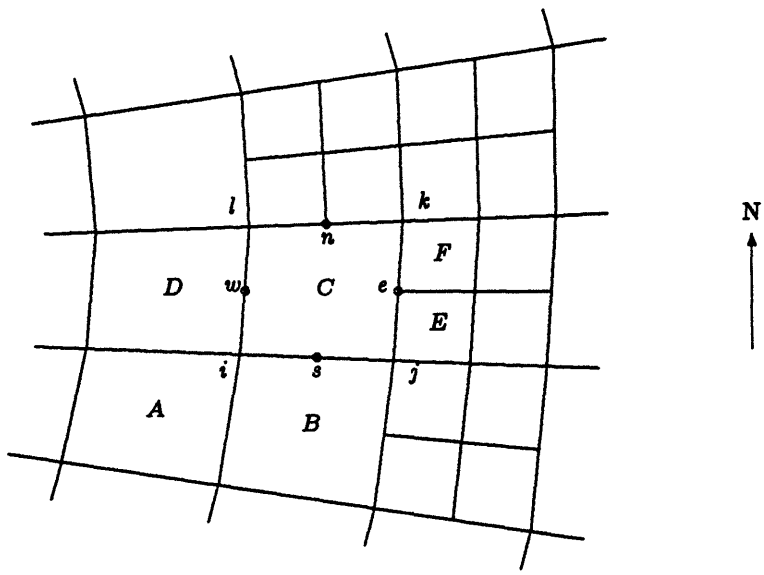

Figure 3.5: Physical grid for flux balance.

While for cell C the nodes e and n actually exist, the nodes w and s are irrelevant; nevertheless for the sake of generalizing the above face fluxes for *all* cells one can define the flux for middle edge nodes for those edges which are not spatial interfaces to be the average of the corresponding corner nodes, for example,

$$F_w = \frac{F_i + F_l}{2}$$

in which case

$$F_W = \frac{F_i + 2F_w + F_l}{4}.$$

Although this may seem to be a trivial point, the above formulation significantly reduces the number of if-then clauses in the actual coding of the solution scheme which involves spatial adaptation.

Statement: *The cell change can be obtained either from a cell in the physical grid or from a corresponding cell in the computational grid (with $\Delta\tilde{U}_C = \Delta U_C/J$), provided that the metrics are given by Equations (3.49) and (3.50).*

Proof:

Let us first consider the cell C in the physical domain. The flux balance is obtained by the trapezoidal integration of Equation (3.6) and is as follows

$$\begin{aligned} \frac{A_C}{\Delta t_C}\Delta U_C \;=\; A_C W_C^* \;&+\; F_W(y_l - y_i) \;-\; G_W(x_l - x_i) \\ &+\; F_N(y_k - y_l) \;-\; G_N(x_k - x_l) \\ &+\; F_E(y_j - y_k) \;-\; G_E(x_j - x_k) \\ &+\; F_S(y_i - y_j) \;-\; G_S(x_i - x_j). \end{aligned} \tag{3.44}$$

If the dependent variables at the middle edge nodes of spatial interfaces are regarded to be the average values of the corresponding corner node values at *all* times, then the flux balance based upon just the corner nodes is appropriate. However, if the changes in dependent variables at the middle nodes of spatial interfaces are computed through some other means, then the inclusion of the middle nodes in the flux balance would yield a more accurate trapezoidal integration [33]. The flux balance using just the corner nodes yields

$$\begin{aligned} \frac{A_C}{\Delta t_C}\Delta U_C \;=\; A_C W_C^* \;&+\; 0.5\,(F_i + F_l)(y_l - y_i) \;-\; 0.5\,(G_i + G_l)(x_l - x_i) \\ &+\; 0.5\,(F_l + F_k)(y_k - y_l) \;-\; 0.5\,(G_l + G_k)(x_k - x_l) \\ &+\; 0.5\,(F_k + F_j)(y_j - y_k) \;-\; 0.5\,(G_k + G_j)(x_j - x_k) \\ &+\; 0.5\,(F_j + F_i)(y_i - y_j) \;-\; 0.5\,(G_j + G_i)(x_i - x_j). \end{aligned} \tag{3.45}$$

This can be rearranged to

$$\begin{aligned} \Delta U_C = \Delta t_C W_C^* + \frac{\Delta t_C}{2A_C}\{ \quad &(F_i - F_k)(y_l - y_j) \;-\; (G_i - G_k)(x_l - x_j) \\ &+\; (F_l - F_j)(y_k - y_i) \;-\; (G_l - G_j)(x_k - x_i) \;\}. \end{aligned} \tag{3.46}$$

The flux balance in the computational coordinates is given by Equation (3.7), *i.e.*,

$$\frac{\Delta\tilde{U}_C}{\Delta t_C} \;=\; \tilde{W}_C^* + \frac{1}{2\Delta\xi}\left\{\tilde{F}_i + \tilde{F}_l - \tilde{F}_j - \tilde{F}_k\right\} - \frac{1}{2\Delta\eta}\left\{\tilde{G}_k + \tilde{G}_l - \tilde{G}_i - \tilde{G}_j\right\}. \tag{3.47}$$

Substituting the values of $\tilde{F}$ and $\tilde{G}$ from Equation (2.67) yields

$$\begin{aligned}\frac{\Delta\tilde{U}_C}{\Delta t_C} = \tilde{W}_C^* &+ \frac{1}{2\Delta\xi}\{\ (y_\eta F - x_\eta G)_i + (y_\eta F - x_\eta G)_l \\ &\quad -(y_\eta F - x_\eta G)_j - (y_\eta F - x_\eta G)_k\ \} \\ &+ \frac{1}{2\Delta\eta}\{\ (x_\xi G - y_\xi F)_i + (x_\xi G - y_\xi F)_j \\ &\quad -(x_\xi G - y_\xi F)_k - (x_\xi G - y_\xi F)_l\ \}. \end{aligned} \tag{3.48}$$

For y_η at node i the forward difference will be used

$$y_{\eta_i} = \frac{y_l - y_i}{\eta_l - \eta_i} = \frac{y_l - y_i}{\Delta\eta}$$

whereas for y_η at node l the backward difference will be used

$$y_{\eta_l} = \frac{y_l - y_i}{\eta_l - \eta_i} = \frac{y_l - y_i}{\Delta\eta}.$$

Thus all the η-derivatives at the corner nodes of cell C are defined as

$$\begin{aligned} y_{\eta_i} &= y_{\eta_l} = \tfrac{y_l - y_i}{\Delta\eta}, & x_{\eta_i} &= x_{\eta_l} = \tfrac{x_l - x_i}{\Delta\eta} \\ y_{\eta_j} &= y_{\eta_k} = \tfrac{y_k - y_j}{\Delta\eta}, & x_{\eta_j} &= x_{\eta_k} = \tfrac{x_k - x_j}{\Delta\eta}. \end{aligned} \tag{3.49}$$

Similarly all the ξ-derivatives of the metrics can be defined as

$$\begin{aligned} x_{\xi_i} &= x_{\xi_j} = \tfrac{x_j - x_i}{\Delta\xi}, & y_{\xi_i} &= y_{\xi_j} = \tfrac{y_j - y_i}{\Delta\xi} \\ x_{\xi_k} &= x_{\xi_l} = \tfrac{x_k - x_l}{\Delta\xi}, & y_{\xi_k} &= y_{\xi_l} = \tfrac{y_k - y_l}{\Delta\xi}. \end{aligned} \tag{3.50}$$

Substituting Equations (3.49) and (3.50) in Equation (3.48) results in

$$\begin{aligned}\frac{\Delta\tilde{U}_C}{\Delta t_C} = \tilde{W}_C^* + \frac{1}{2\Delta\xi\Delta\eta}\{\ &(F_i - F_k)(y_l - y_j) - (G_i - G_k)(x_l - x_j) \\ &+ (F_l - F_j)(y_k - y_i) - (G_l - G_j)(x_k - x_i)\ \}. \end{aligned} \tag{3.51}$$

Using Equation (2.67), this can be reverted back to the physical grid coordinates

$$\begin{aligned}\Delta U_C = \Delta t_C W_C^* + \frac{J\Delta t_C}{2\Delta\xi\Delta\eta}\{\ &(F_i - F_k)(y_l - y_j) - (G_i - G_k)(x_l - x_j) \\ &+ (F_l - F_j)(y_k - y_i) - (G_l - G_j)(x_k - x_i)\ \} \end{aligned} \tag{3.52}$$

This equation is the same as Equation(3.46) if one can show that $J = \Delta\xi\Delta\eta/A_C$. The

metrics for the cell C itself can be defined as

$$\begin{aligned}
x_\xi|_C &= \frac{1}{\Delta\xi}\left(\frac{x_k+x_j}{2}-\frac{x_i+x_l}{2}\right) = \frac{1}{2\Delta\xi}(x_k+x_j-x_i-x_l) = \frac{\Delta x_{cw}}{\Delta\xi} \\
y_\xi|_C &= \frac{1}{\Delta\xi}\left(\frac{y_k+y_j}{2}-\frac{y_i+y_l}{2}\right) = \frac{1}{2\Delta\xi}(y_k+y_j-y_i-y_l) = \frac{\Delta y_{cw}}{\Delta\xi} \\
x_\eta|_C &= \frac{1}{\Delta\eta}\left(\frac{x_l+x_k}{2}-\frac{x_i+x_j}{2}\right) = \frac{1}{2\Delta\eta}(x_l+x_k-x_i-x_j) = \frac{\Delta x_{ns}}{\Delta\eta} \\
y_\eta|_C &= \frac{1}{\Delta\eta}\left(\frac{y_l+y_k}{2}-\frac{y_i+y_j}{2}\right) = \frac{1}{2\Delta\eta}(y_l+y_k-y_i-y_j) = \frac{\Delta y_{ns}}{\Delta\eta}.
\end{aligned} \tag{3.53}$$

Substituting these values in the definition of the Jacobian J yields

$$\frac{1}{J_C} = [x_\xi y_\eta - y_\xi x_\eta]_C = \frac{1}{2\Delta\xi\Delta\eta}[(x_k-x_i)(y_l-y_j)-(x_l-x_j)(y_k-y_i)] = \frac{A_C}{\Delta\xi\Delta\eta}. \tag{3.54}$$

Note that the cell area is one half the cross product of the diagonal vectors of the cell. The Jacobian J for cell C is related to the magnification of the area under the transformation. Thus it has been established that in order for the flux balance to remain valid in both the coordinate systems, the metrics must be defined by Equations (3.49) and (3.50). Q.E.D.

In summary, the flux balance for the cells surrounding the node i is given by

$$\begin{aligned}
\Delta\tilde{U}_A &= \tilde{W}_A\Delta t_A + \frac{\Delta t_A}{2\Delta\xi}\left\{\tilde{F}_a+\tilde{F}_h-\tilde{F}_b-\tilde{F}_i\right\} + \frac{\Delta t_A}{2\Delta\eta}\left\{\tilde{G}_a+\tilde{G}_b-\tilde{G}_h-\tilde{G}_i\right\} \\
\Delta\tilde{U}_B &= \tilde{W}_B\Delta t_B + \frac{\Delta t_B}{2\Delta\xi}\left\{\tilde{F}_b+\tilde{F}_i-\tilde{F}_c-\tilde{F}_j\right\} + \frac{\Delta t_B}{2\Delta\eta}\left\{\tilde{G}_b+\tilde{G}_c-\tilde{G}_i-\tilde{G}_j\right\} \\
\Delta\tilde{U}_C &= \tilde{W}_C\Delta t_C + \frac{\Delta t_C}{2\Delta\xi}\left\{\tilde{F}_i+\tilde{F}_l-\tilde{F}_j-\tilde{F}_k\right\} + \frac{\Delta t_C}{2\Delta\eta}\left\{\tilde{G}_i+\tilde{G}_j-\tilde{G}_l-\tilde{G}_k\right\} \\
\Delta\tilde{U}_D &= \tilde{W}_D\Delta t_D + \frac{\Delta t_D}{2\Delta\xi}\left\{\tilde{F}_h+\tilde{F}_m-\tilde{F}_i-\tilde{F}_l\right\} + \frac{\Delta t_D}{2\Delta\eta}\left\{\tilde{G}_h+\tilde{G}_i-\tilde{G}_m-\tilde{G}_l\right\}.
\end{aligned} \tag{3.55}$$

As asserted earlier the time-step in these cell changes is assumed to be constant. The average of these cell changes can be denoted by ΔU_i and it will be shown to be the FOCIT at node i

$$\begin{aligned}
\Delta\tilde{U}_i = \frac{\Delta t}{4}(\tilde{W}_A+\tilde{W}_B+\tilde{W}_C+\tilde{W}_D) &+ \frac{\Delta t}{2\Delta\xi}\left\{\frac{\tilde{F}_a+2\tilde{F}_h+\tilde{F}_m}{4}-\frac{\tilde{F}_c+2\tilde{F}_j+\tilde{F}_k}{4}\right\} \\
&+\frac{\Delta t}{2\Delta\eta}\left\{\frac{\tilde{G}_a+2\tilde{G}_b+\tilde{G}_c}{4}-\frac{\tilde{G}_m+2\tilde{G}_l+\tilde{G}_k}{4}\right\}.
\end{aligned} \tag{3.56}$$

It is seen that the first curly bracket represents $-2\Delta\xi\frac{\partial\tilde{F}}{\partial\xi}$ at node i with weighting factors as indicated in Figure (3.6). Similarly the second curly bracket represents $-2\Delta\eta\frac{\partial\tilde{G}}{\partial\eta}$ at the common node.

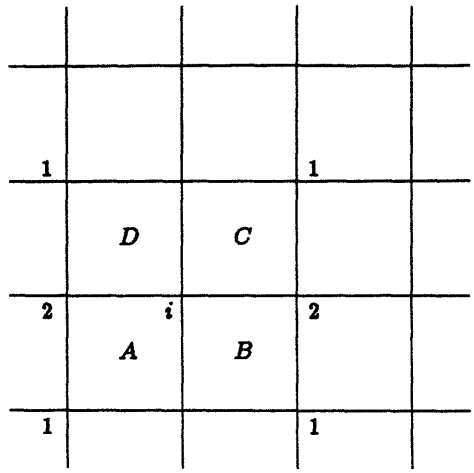

Figure 3.6: Nodes used for the computation of $\partial F/\partial \xi$. The numerals are the weighting factors for the nodes.

Defining the source term at node i as the average

$$\tilde{W}_i = \frac{1}{4}\left(\tilde{W}_A + \tilde{W}_B + \tilde{W}_C + \tilde{W}_D\right). \tag{3.57}$$

Equation (3.56) can be written as

$$\text{FOCIT}_i = \Delta\tilde{U}_i = \Delta t\left(\tilde{W}_i + \frac{\partial \tilde{F}}{\partial \xi} + \frac{\partial \tilde{G}}{\partial \eta}\right)_i = \frac{1}{4}\left(\Delta\tilde{U}_A + \Delta\tilde{U}_B + \Delta\tilde{U}_C + \Delta\tilde{U}_D\right). \tag{3.58}$$

But this is recognized to be the first order change at node i. The second order source term in SOCIT (Eq. 3.42) is given by the average

$$q\frac{\Delta t^2}{2}\tilde{W}_U\tilde{U}_t\Big|_i = q\frac{\Delta t^2}{2}\frac{1}{4}\left[\tilde{W}_U\frac{\partial\tilde{U}}{\partial t}\Big|_A + \tilde{W}_U\frac{\partial\tilde{U}}{\partial t}\Big|_B + \tilde{W}_U\frac{\partial\tilde{U}}{\partial t}\Big|_C + \tilde{W}_U\frac{\partial\tilde{U}}{\partial t}\Big|_D\right]$$

here $\tilde{W}_U$ is used as a simplified notation for $\frac{\partial\tilde{W}}{\partial\tilde{U}}$; since $\Delta\tilde{U}_A = \tilde{U}_{t_A}\Delta t$ *etc.*, this gives

$$q\frac{\Delta t^2}{2}\tilde{W}_U\tilde{U}_t\Big|_i = q\frac{\Delta t}{8}\left[\Delta\tilde{W}_A + \Delta\tilde{W}_B + \Delta\tilde{W}_C + \Delta\tilde{W}_D\right] \tag{3.59}$$

where the ***source change*** is given by, for example,

$$\Delta\tilde{W}_C = \frac{\partial\tilde{W}}{\partial\tilde{U}}\Big|_C \Delta\tilde{U}_C = \frac{1}{J}\frac{\partial W}{\partial U}\Big|_C \Delta U_C. \tag{3.60}$$

The second order F-flux term in SOCIT is

$$-\frac{\Delta t^2}{2}\frac{\partial}{\partial \xi}\left(\tilde{F}_U\tilde{U}_t\right)_i = -\frac{\Delta t^2}{2\Delta \xi}\left[\frac{1}{2}\left(\tilde{F}_U\frac{\partial \tilde{U}}{\partial t}\bigg|_A + \tilde{F}_U\frac{\partial \tilde{U}}{\partial t}\bigg|_D\right) - \frac{1}{2}\left(\tilde{F}_U\frac{\partial \tilde{U}}{\partial t}\bigg|_B + \tilde{F}_U\frac{\partial \tilde{U}}{\partial t}\bigg|_C\right)\right].$$

Again using $\tilde{U}_t\Delta t$ as a value for $\Delta\tilde{U}$ implies

$$\begin{aligned}-\frac{\Delta t^2}{2}\frac{\partial}{\partial \xi}\left(\tilde{F}_U\tilde{U}_t\right)_i &= -\frac{\Delta t}{4\Delta \xi}\left[\tilde{F}_U\Delta\tilde{U}\big|_A + \tilde{F}_U\Delta\tilde{U}\big|_D - \tilde{F}_U\Delta\tilde{U}\big|_B - \tilde{F}_U\Delta\tilde{U}\big|_C\right] \\ &= -\frac{\Delta t}{4\Delta \xi}\left[\Delta\tilde{F}_A - \Delta\tilde{F}_B - \Delta\tilde{F}_C + \Delta\tilde{F}_D\right]. \qquad (3.61)\end{aligned}$$

Similarly the second order G-flux term in SOCIT is

$$-\frac{\Delta t^2}{2}\frac{\partial}{\partial \eta}\left(\tilde{G}_U\tilde{U}_t\right)_i = -\frac{\Delta t}{4\Delta \eta}\left[\Delta\tilde{G}_A + \Delta\tilde{G}_B - \Delta\tilde{G}_C - \Delta\tilde{G}_D\right] \qquad (3.62)$$

where the *Jacobian changes* are given by

$$\begin{aligned}\Delta\tilde{F} &= \left(y_\eta\frac{\partial F}{\partial U} - x_\eta\frac{\partial G}{\partial U}\right)\Delta U = y_\eta\Delta F - x_\eta\Delta G \\ \Delta\tilde{G} &= \left(x_\xi\frac{\partial G}{\partial U} - y_\xi\frac{\partial F}{\partial U}\right)\Delta U = x_\xi\Delta G - y_\xi\Delta F.\end{aligned} \qquad (3.63)$$

These values for cell C, using Equation (3.53), are

$$\begin{aligned}\Delta\tilde{F}_C &= \frac{\Delta y_{ns}\Delta F - \Delta x_{ns}\Delta G}{\Delta\eta} \\ \Delta\tilde{G} &= \frac{\Delta x_{ew}\Delta G - \Delta y_{ew}\Delta F}{\Delta\xi}\end{aligned} \qquad (3.64)$$

where, $\Delta F = F_U\Delta U$, *etc.*

Now adding the three terms contributing to SOCIT (*i.e.*, substituting Eqs. 3.59, 3.61 and 3.62 in 3.42) yields

$$\begin{aligned}\text{SOCIT} = \frac{1}{2}\frac{\partial^2\tilde{U}}{\partial t^2}\bigg|_i^n\Delta t^2 &= q\frac{\Delta t}{8}\left(\Delta\tilde{W}_A + \Delta\tilde{W}_B + \Delta\tilde{W}_C + \Delta\tilde{W}_D\right) \\ &+\frac{\Delta t}{4\Delta\xi}\left(\Delta\tilde{F}_A - \Delta\tilde{F}_B - \Delta\tilde{F}_C + \Delta\tilde{F}_D\right) \\ &+\frac{\Delta t}{4\Delta\eta}\left(\Delta\tilde{G}_A + \Delta\tilde{G}_B - \Delta\tilde{G}_C - \Delta\tilde{G}_D\right). \qquad (3.65)\end{aligned}$$

The substitution of Equation (3.58) and the above equation in Equation (3.40) yields the discretized version of the change at node i without artificial damping, *viz.*

$$\begin{aligned}4\delta\tilde{U}_i &= \left(\Delta\tilde{U}_A + \tfrac{\Delta t_A}{\Delta\xi}\Delta\tilde{F}_A + \tfrac{\Delta t_A}{\Delta\eta}\Delta\tilde{G}_A + \tfrac{q}{2}\Delta t_A\Delta\tilde{W}_A\right) + \\ &\left(\Delta\tilde{U}_B - \tfrac{\Delta t_B}{\Delta\xi}\Delta\tilde{F}_B + \tfrac{\Delta t_B}{\Delta\eta}\Delta\tilde{G}_B + \tfrac{q}{2}\Delta t_B\Delta\tilde{W}_B\right) + \\ &\left(\Delta\tilde{U}_C - \tfrac{\Delta t_C}{\Delta\xi}\Delta\tilde{F}_C - \tfrac{\Delta t_C}{\Delta\eta}\Delta\tilde{G}_C + \tfrac{q}{2}\Delta t_C\Delta\tilde{W}_C\right) + \\ &\left(\Delta\tilde{U}_D + \tfrac{\Delta t_D}{\Delta\xi}\Delta\tilde{F}_D - \tfrac{\Delta t_D}{\Delta\eta}\Delta\tilde{G}_D + \tfrac{q}{2}\Delta t_D\Delta\tilde{W}_D\right).\end{aligned} \qquad (3.66)$$

The overall change δU_i in the previous equation may be thought of as contributions from cells A-through D, *i.e.*,

$$\delta\tilde{U}_i \;=\; \delta\tilde{U}_{iA} + \delta\tilde{U}_{iB} + \delta\tilde{U}_{iC} + \delta\tilde{U}_{iD}. \tag{3.67}$$

These values are given by

$$\begin{aligned}
\delta\tilde{U}_{iA} &= \tfrac{1}{4}\left[\Delta\tilde{U}_A + \tfrac{\Delta t_A}{\Delta\xi}\Delta\tilde{F}_A + \tfrac{\Delta t_A}{\Delta\eta}\Delta\tilde{G}_A + q\tfrac{\Delta t_A}{2}\Delta\tilde{W}_A\right]\\
\delta\tilde{U}_{iB} &= \tfrac{1}{4}\left[\Delta\tilde{U}_B - \tfrac{\Delta t_B}{\Delta\xi}\Delta\tilde{F}_B + \tfrac{\Delta t_B}{\Delta\eta}\Delta\tilde{G}_B + q\tfrac{\Delta t_B}{2}\Delta\tilde{W}_B\right]\\
\delta\tilde{U}_{iC} &= \tfrac{1}{4}\left[\Delta\tilde{U}_C - \tfrac{\Delta t_C}{\Delta\xi}\Delta\tilde{F}_C - \tfrac{\Delta t_C}{\Delta\eta}\Delta\tilde{G}_C + q\tfrac{\Delta t_C}{2}\Delta\tilde{W}_C\right]\\
\delta\tilde{U}_{iD} &= \tfrac{1}{4}\left[\Delta\tilde{U}_D + \tfrac{\Delta t_D}{\Delta\xi}\Delta\tilde{F}_D - \tfrac{\Delta t_D}{\Delta\eta}\Delta\tilde{G}_D + q\tfrac{\Delta t_D}{2}\Delta\tilde{W}_D\right].
\end{aligned} \tag{3.68}$$

It is now possible to write down the contributions of any cell to its corner nodes. Specifically for cell C the distribution relations in computational coordinates is given by

$$\begin{aligned}
\delta\tilde{U}_{iC} &= \tfrac{1}{4}\left[\Delta\tilde{U}_C - \tfrac{\Delta t_C}{\Delta\xi}\Delta\tilde{F}_C - \tfrac{\Delta t_C}{\Delta\eta}\Delta\tilde{G}_C + q\tfrac{\Delta t_C}{2}\Delta\tilde{W}_C\right]\\
\delta\tilde{U}_{jC} &= \tfrac{1}{4}\left[\Delta\tilde{U}_C + \tfrac{\Delta t_C}{\Delta\xi}\Delta\tilde{F}_C - \tfrac{\Delta t_C}{\Delta\eta}\Delta\tilde{G}_C + q\tfrac{\Delta t_C}{2}\Delta\tilde{W}_C\right]\\
\delta\tilde{U}_{kC} &= \tfrac{1}{4}\left[\Delta\tilde{U}_C + \tfrac{\Delta t_C}{\Delta\xi}\Delta\tilde{F}_C + \tfrac{\Delta t_C}{\Delta\eta}\Delta\tilde{G}_C + q\tfrac{\Delta t_C}{2}\Delta\tilde{W}_C\right]\\
\delta\tilde{U}_{lC} &= \tfrac{1}{4}\left[\Delta\tilde{U}_C - \tfrac{\Delta t_C}{\Delta\xi}\Delta\tilde{F}_C + \tfrac{\Delta t_C}{\Delta\eta}\Delta\tilde{G}_C + q\tfrac{\Delta t_C}{2}\Delta\tilde{W}_C\right].
\end{aligned} \tag{3.69}$$

Substituting Equations (2.67) and (3.64) in these distribution relations yields the corresponding relations in physical coordinates, *viz.*

$$\begin{aligned}
\delta U_{iC} &= \tfrac{1}{4}\left[\Delta U - \tfrac{\Delta t}{A}\left(\Delta y_{ns}\Delta F - \Delta x_{ns}\Delta G\right) - \tfrac{\Delta t}{A}\left(\Delta x_{ew}\Delta G - \Delta y_{ew}\Delta F\right) + q\tfrac{\Delta t}{2}\Delta W + \Psi_i\right]_C\\
\delta U_{jC} &= \tfrac{1}{4}\left[\Delta U + \tfrac{\Delta t}{A}\left(\Delta y_{ns}\Delta F - \Delta x_{ns}\Delta G\right) - \tfrac{\Delta t}{A}\left(\Delta x_{ew}\Delta G - \Delta y_{ew}\Delta F\right) + q\tfrac{\Delta t}{2}\Delta W + \Psi_j\right]_C\\
\delta U_{kC} &= \tfrac{1}{4}\left[\Delta U + \tfrac{\Delta t}{A}\left(\Delta y_{ns}\Delta F - \Delta x_{ns}\Delta G\right) + \tfrac{\Delta t}{A}\left(\Delta x_{ew}\Delta G - \Delta y_{ew}\Delta F\right) + q\tfrac{\Delta t}{2}\Delta W + \Psi_k\right]_C\\
\delta U_{lC} &= \tfrac{1}{4}\left[\Delta U - \tfrac{\Delta t}{A}\left(\Delta y_{ns}\Delta F - \Delta x_{ns}\Delta G\right) + \tfrac{\Delta t}{A}\left(\Delta x_{ew}\Delta G - \Delta y_{ew}\Delta F\right) + q\tfrac{\Delta t}{2}\Delta W + \Psi_l\right]_C.
\end{aligned} \tag{3.70}$$

Here the term Ψ incorporates the effect of artificial viscosity which will be described separately in a later section. These distribution formulae allow for different time-steps and cell volumes for cells adjoining a common node. Starting with zero changes at all nodes, these distribution formulae allow one to integrate on a cell by cell basis and hence

accumulate changes at the corner nodes by summing the current contributions to the already existing values at the nodes due to the previous integrations on the neighboring cells. Once all the cells are integrated the nodes can be updated and reset to zero change values again. The terms Δx and Δy are as defined by Equation (3.53), *i.e.*, for example

$$\Delta x_{ew} = \frac{1}{2}(x_k + x_j - x_i - x_l).$$

The strategy when treating the source term implicitly, *i.e.*, choosing $W^* = W^{n+1}$ for a cell in Equation (3.44), requires discussion. Stability analysis of a linearized source term model, to be discussed in the next chapter, shows that no substantial gain in stability limits is acquired, over the explicit scheme, if the second order source term is retained while treating the first order source term implicitly. However, if only first order implicit source terms are retained ($q = 0$ in Eq. 3.70) the stability of the model equation becomes independent of the magnitude of the source term and is constrained solely by the familiar CFL condition. Therefore, for a system of equations it is reasonable to use the $q = 0$ simplification with a source implicit scheme, and $q = 1$ with an explicit scheme.

The implicit source term for a cell C can be approximated by Newton linearization

$$W_C^{n+1} = W_C^n + \left.\frac{\partial W}{\partial U}\right|_C \Delta U_C. \tag{3.71}$$

On substituting this in Equation (3.6) the following is obtained

$$\Delta U_C = \left(I - \left.\frac{\partial W}{\partial U}\right|_C \Delta t_C\right)^{-1} \left[\Delta t_C W_C^n + \frac{\Delta t_C}{A_C}\oint_C (F dy - G dx)\right]. \tag{3.72}$$

The corresponding discretized version is obtained by substituting Equation (3.44) into the square bracket

$$\Delta U_C^{IM} = \left(I - \left.\frac{\partial W}{\partial U}\right|_C \Delta t_C\right)^{-1} \Delta U_C^{EX}. \tag{3.73}$$

The superscripts emphasize the relationship between the implicit and explicit cell changes. The matrix premultiplying the explicit cell change is often referred to as the ***preconditioning matrix.*** This equation is used in conjunction with the distribution formulae (Eq. 3.70 with $q = 0$) while looping over cells whenever a source implicit scheme is used instead of Equation (3.44). The source implicit scheme reduces to the explicit

scheme for the non-reacting case. An alternative way is to compute the cell changes explicitly but use the preconditioning matrix on the distribution formulae. This approach is elaborated in the next chapter. It must be emphasized again that the implicit source vector may be used to overcome the severe time-step restriction imposed by the otherwise *stiff* chemical systems but not to by-pass the time resolution requirements which may be necessary to capture the inherent physics of the reactions. A discussion of the resolution time requirements appears in the Chapter 6 of temporal adaptation.

3.5 Spatial Interface Treatment

As mentioned earlier, the introduction of embedded regions into an otherwise coarse mesh leads to the formation of spatial interfaces which must be treated so as to yield stable and accurate results. Two alternative procedures have been considered for the middle edge node of a spatial interface. In the first approach node e is handled in the usual manner (Eq. 3.70) when integrating cells E and F in Figure (3.5). When integrating C, a simple average is used for the change at e, *i.e.*,

$$\delta U_{eC} = \frac{\delta U_{jC} + \delta U_{kC}}{2}.$$

In this approach the contribution of cell C to the changes at the corner nodes involves a flux balance which takes into account the hanging nodes; *e.g.*, the east F-flux is $F_E = (F_j + 2F_e + F_k)/4$. Hence, in the absence of temporal adaptation, the total change accumulated at node e once all of the cells are integrated is

$$\delta U_e = \delta U_{eC} + \delta U_{eE} + \delta U_{eF}.$$

This approach is tantamount to performing a special integration over the spatial interface as demonstrated by Dannenhoffer [33]. It will be referred to as the *average change approach for spatial interface.*

The second approach determines the value of the state vector at the middle edge node by interpolating from the corresponding corner node values. Since by construction the middle edge nodes form the midpoints of the corresponding corner nodes of the spatial

interface, a second order interpolation implies that the state vector at this node is equal to the average of the corner nodes at *all* times, thus for example,

$$U_e = \frac{U_j + U_k}{2}.$$

In this non-conservative approach only corner nodes are involved in the flux balance for any cell integration. Hence when cells E and F are integrated, the changes at node e are accumulated in the usual manner, while the change from cell C at node e would not be included. When updating of the nodes, node e will be recognized to be a middle edge node and its state will be set according to the previous equation, thereby making the accumulation of changes at node e due to cells E and F irrelevant. This approach will be referred to as the *average state vector approach for spatial interface.* This approach had been utilized by Usab [133].

The results for the two approaches yield identical graphical output for most cases. The second approach is simpler, involves no if-then clauses for the flow solver except at the time of updating, and hence can be easily vectorized. Furthermore this approach can be extended easily to 3-D and would be suitable for new kinds of interfaces; *e.g.*, those generated by *directional embedding* [71]. However, due to the non-conservative nature of the approach, care must be exercized in moving the interfaces away from the actual shock locations. This can be achieved by adding buffer zones to the spatially resolved region. Due to the robustness of the second approach, it was decided to base the solver on that approach in the latest version of STAR code.

3.6 Artificial Viscosity in Two Spatial Dimensions

The generalization of the 1-D modified differential equation (Eq. 3.24) to two spatial dimensions is

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = W + \frac{\sigma}{4\Delta s}\left(\frac{\partial^2 U}{\partial x^2}\Delta x^2 + \frac{\partial^2 U}{\partial y^2}\Delta y^2\right) \tag{3.74}$$

where Δs is some typical cell dimension which will be evaluated later. Considering a five point stencil comprising of the cell centers about node i in Figure (3.5) the Laplacian

type terms can be written as

$$\frac{\partial^2 U}{\partial x^2}\Delta x^2+\frac{\partial^2 U}{\partial y^2}\Delta y^2 = \left(\frac{U_C+U_B}{2}-2U_i+\frac{U_A+U_D}{2}\right)+\left(\frac{U_C+U_D}{2}-2U_i+\frac{U_A+U_B}{2}\right). \tag{3.75}$$

This can be rearranged to give

$$\frac{\partial^2 U}{\partial x^2}\Delta x^2+\frac{\partial^2 U}{\partial y^2}\Delta y^2 = (U_A-U_i)+(U_B-U_i)+(U_C-U_i)+(U_D-U_i). \tag{3.76}$$

Thus the contribution of artificial viscosity from cell C to node i is

$$\delta U^*_{iC} = \frac{\sigma\Delta t_C}{4\Delta s}(U_C-U_i) \tag{3.77}$$

and hence the Ψ term in Equation (3.70) is

$$\Psi_{iC} = \frac{\sigma\Delta t_C}{\Delta s_C}(U_C-U_i). \tag{3.78}$$

If the artificial viscosity coefficient is allowed to vary with the nodes then a non-convective, conservative formulation would imply

$$\Psi_{iC} = \frac{\Delta t_C}{\Delta s_C}[(\sigma U)_C-\sigma_i U_i] \tag{3.79}$$

where

$$(\sigma U)_C = \frac{1}{4}(\sigma_i U_i+\sigma_j U_j+\sigma_k U_k+\sigma_l U_l). \tag{3.80}$$

Ni had taken the dimension Δs to be

$$\frac{1}{\Delta s} = \frac{1}{\Delta x}+\frac{1}{\Delta y}. \tag{3.81}$$

Thus Δs is proportional to the harmonic mean of the two linear dimensions of a rectangular cell. For a general quadrilateral cell these dimensions are ambiguous, therefore the following measure is proposed

$$\Delta s = \frac{4A_C}{P_C} \tag{3.82}$$

where the denominator represents the perimeter of the cell. Note that this relation implies Δs to be the harmonic mean of Δx and Δy and two times the value proposed by Ni. If this factor of two is absorbed in the viscosity coefficient itself then the viscosity here should be twice as large as Ni's viscosity coefficient to produce the same level

of artificial diffusion. Also note that for very high aspect ratio cells the dimension Δs will approximately scale as two times the minimum dimension and hence would correspondingly imply a larger value of dissipation.

In line with the approach utilized for 1-D, normalized first differences of density are used for evaluating the artificial viscosity, which then is stored at all nodes and has the general form

$$\sigma_i = \sigma_{min} + \frac{\delta}{4}(\kappa_A + \kappa_B + \kappa_C + \kappa_D) \tag{3.83}$$

where, κ_C, for example, is the normalized scaling which is a combination of density differences along the two cell dimensions, *i.e.*,

$$\kappa_C = \left|\frac{\rho_e - \rho_w}{\rho_e + \rho_w}\right| + \left|\frac{\rho_n - \rho_s}{\rho_n + \rho_s}\right|. \tag{3.84}$$

For cells A and D where the edge nodes do not appear, average values of the corresponding corner nodes are used for evaluating the scalings. The constant δ is chosen so that $\sigma \in [\sigma_{min}, \sigma_{max}]$, typically between 0.05 and 0.5.

In the present algorithm artificial viscosity is introduced only at the corner nodes whenever integrating a particular cell. This is true without qualifications when the average state vector approach is used for handling spatial interfaces. For the average change approach, it has been experimentally observed that for a node such as e a lower viscosity coefficient is needed. Hence a natural way of accumulating artificial viscosity at such a node is to use Equation (3.83) but only for cells whose corner is e, *i.e.*,

$$\sigma_e = \sigma_{min} + \frac{\delta}{4}(\kappa_E + \kappa_F).$$

A plausible reason that lesser artificial viscosity is needed at middle edge nodes is that the changes at the corner nodes of the larger cell already account for artificial viscosity at this node. In particular for cell C in Figure (3.5), the node e has the change $(\delta U_{jC} + \delta U_{kC})/2$ and each of these corner changes have contributions from artificial viscosity and its value from cell C is $(\Psi_{jC} + \Psi_{kC})/2$; hence additional artificial viscosity from cell C is not needed. However, the artificial viscosity from cells E and F involves a flux balance and hence requires explicit addition of smoothing.

In order to avoid unnecessary if-then clauses, the above formula can also be used for the average state vector approach at the middle edge nodes. Since the changes at these nodes are irrelevant for this approach, the actual artificial viscosity coefficient at such nodes is also of no consequence.

For ease of application in coding and vectorization considerations the following procedure is proposed for the determination of artificial viscosity coefficient at all the nodes:

1. March over all nodes i and set

$$\sigma_i := \sigma_{min} , \qquad i = 1, \ldots, N_n$$

where N_n is the total number of nodes. The notation := is used here to emphasize computer assignment.

2. March over cells c and sum up the contributions from individual cells over the corner nodes

$$\sigma_i := \sigma_i + \frac{\delta}{4}\kappa_c , \qquad i = 1, \ldots, N_c$$

where N_c is the total number of cells and i in the above assignment is a corner node of some cell c. Hence for cell C in Figure (3.5) this assignment will loop over nodes i, j, k, l. In this expression κ_C is computed from Equation (3.84) and δ has the value assigned from the previous invocation of this procedure. For initialization purposes δ can be set equal to zero and its value can be determined by the following step; subsequently the procedure can be called again to have the correct assignment of artificial viscosity at the nodes.

3. The march over cells also determines κ_{max} to be the maximum value of all κ_c, *i.e.*,

$$\kappa_{max} := \max\{\kappa_c\} , \qquad c = 1, \ldots, N_c.$$

For the given values of minimum and maximum artificial viscosity coefficient, *viz.*, σ_{min} and σ_{max}, the value of the constant δ can be determined as

$$\delta := \frac{\sigma_{max} - \sigma_{min}}{\kappa_{max}}.$$

This expression will approximately keep the artificial viscosity coefficient between σ_{min} and σ_{max}.

4. Finally the boundary nodes b are adjusted by using a reflective condition. This can be accomplished by marching over the boundary nodes and setting

$$\sigma_b := 2\sigma_b - \sigma_{min}\ , \qquad b = 1, \ldots, N_b$$

where N_b is the total number of boundary nodes which border two cells. For boundary nodes which border only one cell this assignment is changed to

$$\sigma_b := 4\sigma_b - 3\sigma_{min}.$$

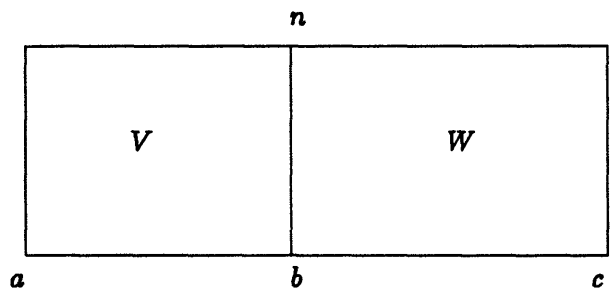

Figure 3.7: Finite volumes adjacent to a boundary node b.

Note that step (3) of this procedure automatically satisfies Equation (3.83) at the usual nodes and the corresponding equation (after Eq. 3.84) at the middle edge nodes. The march over boundary nodes deserves special attention. Consider cells V and W adjacent to a boundary a-b-c of the computational domain as shown in Figure (3.7). The march over nodes and cells of the computational domain yields the following value at node b

$$\sigma_b = \sigma_{min} + \frac{\delta}{4}(\kappa_V + \kappa_W)$$

whereas for node n there are four cell contributions to σ_n, hence for a uniform flow it will be observed that

$$\sigma_n = \sigma_{min} + \delta\kappa_W \qquad \text{and} \qquad \sigma_b = \sigma_{min} + \frac{\delta}{2}\kappa_W.$$

These two expressions can be made consistent if the artificial viscosity at boundary node b is assumed to be summed from cells V and W and their corresponding *reflective*

cells which introduce the same contributions. In other words the wall cell contributions ought to be multiplied by a factor of 2. Thus the corrected value for the boundary node is

$$\sigma_b^c \;=\; \sigma_{min} + \frac{\delta}{2}(\kappa_V + \kappa_W) \;=\; \sigma_{min} + 2(\sigma_b - \sigma_{min}) \;=\; 2\sigma_b - \sigma_{min}. \qquad (3.85)$$

A similar explanation holds for the four corner boundary nodes which border a single cell in the computational domain.

Chapter 4

Stiff Chemical Systems

An important step in the development of a new algorithm is the determination of time-step restrictions through a stability analysis. Even for well-established schemes a stability analysis can provide understanding of the physical domain of dependence. This chapter starts with an introduction to the concept of stiffness followed by a result pertaining to a linear frozen convective wave equation. Section (4.3) explores the origin of stiffness in a one-dimensional model and possible remedy for this phenomenon by a Von-Neumann analysis. Section (4.4) compares the exact solution of a linear differential equation of first order with those from numerical schemes of interest whereas Section (4.5) discusses the implementation of the source implicit scheme for both one and two dimensional situations. The source implicit algorithms are the ones which are implicit only in the source terms and the rest of the terms are modelled explicitly. Finally Section (4.6) describes an alternate method for avoiding chemically stiff reaction systems.

The usual approach for analyzing stability on structured grids makes use of Fourier analysis, which considers a general solution to be a sum of Fourier modes which are amenable to separate analysis. This is frequently referred to as Von Neumann stability analysis. The numerical integration techniques to be considered here are the fully explicit and source implicit methods for the present algorithm.

4.1 Introduction

Stiffness is a numerical phenomenon which is exhibited in complex systems when some components of their solutions respond promptly to system perturbations whereas others respond relatively slowly. The degree of stiffness increases with the widening of these individual responses. The concept of stiffness arises from both the numerics of a given computational scheme and the physical model which it describes. A system of equations describing a transient phenomenon associated with multiple reactions in a closed volume (no convection!) is stiff if the eigenvalues of the Jacobian matrix of the source vector has widely disparate negative real parts. In contrast to stiff problems of this sort there are unstable systems which are characterized by positive eigenvalues and oscillatory systems that have mostly complex eigenvalues. The stiffness pertaining to chemical reactions can be traced back to widely different reaction rates, *i.e.*, fast reactions (large rate coefficients) imply smaller characteristic time scales and *vice versa*. Such large source terms produce rapid temporal changes which can lead to constraints for stable computations. When convective terms are also considered, the eigenvalues of the Jacobians of flux vectors must also be taken into account. The convective eigenvalues can be positive or negative and have no bearing on the stability of the physical model so long as the eigenvalues of the source vector have negative real parts. However, the corresponding computational model usually has stability restrictions based upon the largest magnitude eigenvalue of the flux vector, in addition to the restrictions based upon chemical time-scales.

In general, stiffness is characterized by an enormous difference in eigenvalue magnitudes of the Jacobian matrices, and a measure of stiffness is the magnitude of the ratio of the largest to the smallest eigenvalue. Thus, even when all the reactions in a multi-reaction system proceed at comparable rates, the system of equations can still be stiff if the fluid time-scale is widely disparate from a typical chemical characteristic time-scale. In the description of phenomena like flames, combustion and detonations, the pertinent time-scales can easily range over several orders of magnitude. The simultaneous representation of these diverse time-scales manifests itself as a limitation

in temporal accuracy in the sense that the allowable time-step becomes smaller than the smallest time-scale in the problem. The smallest time-scale of a certain process may or may not be the most important one. For example, if a process relaxes in time and approaches an asymptotic limit, the smallest time-scales are important only during the relaxation phase. If the important time-scales can be resolved by a suitable (*e.g.*, implicit) algorithm, then obtaining a desired temporal accuracy is not necessarily a limitation. However, algorithms requiring advancement on the basis of smallest time-scale, will necessitate computing for a large number of time-steps, thereby making the cost of simulations prohibitive.

Explicit algorithms typically suffer from a stability restriction that requires the allowable time-step to be related to the slowest characteristic time-scale in the problem. Even after the decay of fast transients the solutions vary slowly and the explicit methods can require exceedingly small time-steps to maintain stability. One is either forced to use implicit integration schemes or modify the explicit scheme for a different set of source vectors. Both these techniques will be further explored in subsequent sections of this chapter. These techniques, however, use much more computer resources for each time-step than their explicit counterparts, but have better stability properties and can therefore advance through much larger time-steps. Time-step selection can then be based on accuracy considerations rather than the severe stability restriction.

If the chemical time-scale of a particular reaction is infinitely small compared to those of other reactions everywhere in the spatial domain and at all times, then an equilibrium chemistry model can be utilized for that reaction; however, other chemical reactions must still be modelled by finite rate kinetics. This generally complicates the numerics of the reaction systems because special procedures are required to handle this ***partial equilibrium*** [5,20,112] where only a few reactions are in equilibrium at all times and at all spatial locations. The primitive equations describing the partial equilibrium situations are inconvenient to use because the progress rates Ω_{fr}, Ω_{br} for the equilibrium reactions are determined implicitly from the associated equilibrium constraint conditions. The robustness of computer programs is generally sacrificed due to the addition of special cases which only apply to specific reaction systems. Furthermore the occurrance of

partial equilibrium is infrequent, since the chemical time-scales for a reaction rarely remain infinitely small and constant, both globally (in space) and eternally (in time), compared to those of the other reactions. For most reaction systems the time-scales can vary significantly throughout the domain of interest. Since this variation is generally not known *apriori*, an algorithm must be able to treat a wide range of time-scales.

As a final note to this section it is appropriate to mention recent references Aiken [2], Kee and Dwyer [72] and Oran and Boris [100], which include good discussions of stiffness due to chemical reactions.

4.2 Stability of a 2-D convective wave equation

A Von Neumann stability analysis for a 2-D scalar wave equation of the form

$$U_t + uU_x + vU_y = 0 \tag{4.1}$$

has been performed by Usab [133] for the Ni scheme. In this equation the *characteristic speeds* u and v were regarded as constants. The form of this wave equation is similar to the *decoupled* Euler equations without source terms. Hence the stability limits for the *linearized* Euler equations can be inferred directly from the analysis of the 2-D wave equation. The time-step restriction so obtained is referred to as the CFL condition and is of the form

$$\frac{\Delta t_{CFL}}{\Gamma A_C} \leq \min\left\{ \frac{1}{|u\Delta y_{ns} - v\Delta x_{ns}| + a_f \mathcal{D}_{ns}}, \frac{1}{|u\Delta y_{ew} - v\Delta x_{ew}| + a_f \mathcal{D}_{ew}} \right\} \tag{4.2}$$

here a_f is the local frozen speed of sound for some cell, Γ the CFL number, the cell dimensions Δx and Δy are as defined in Chapter 3 and

$$\mathcal{D}^2 = \Delta x^2 + \Delta y^2. \tag{4.3}$$

The CFL constraint states that the time-step is restricted by requiring the information not to propagate beyond the *domain of dependence* for the two coordinate directions. For the Ni scheme the CFL number must be kept less than unity.

4.3 Stability of a 1-D Scalar Equation with Source Term

In order to study the effect of a source term on stability analysis for Ni scheme consider a simple linear scalar equation of the form

$$\frac{\partial U}{\partial t} + u\frac{\partial U}{\partial x} = \frac{U_e - U}{\tau} \tag{4.4}$$

where u is a characteristic convection speed, τ is the characteristic time-scale for the non-equilibrium process and U_e is the corresponding equilibrium state which the process tries to achieve. The right side of this equation represents a simplistic model for the source term which retains the essential physics of reacting systems and is amenable to analytic study. This equation represents the convection phenomenon and localized processes such as mass source and sink terms, dissipation effects, equilibration in chemical reactions, *etc.* The characteristic time-scale can vary from zero (equilibrium flows) to infinity (frozen flows). Another interpretation, for the time-scale τ, can be presented, if the above equation is compared to individual conservation equations in a reacting system. If U represents the density then this time-scale is infinite, however if it represents the degree of dissociation Y_1 of a relaxing gas then the time-scale will be finite for a non-equilibrium process. For the Lighthill model presented in Chapter 2 this non-dimensional time-scale can be written as

$$\tau = \frac{\rho(Y_{1_e} - Y_1)}{\dot{W}_1} = \frac{Y_{1_e} - Y_1}{\Phi T^\eta \rho \left[(1 - Y_1)e^{-\theta_d/T} - \frac{\rho}{\rho_d}Y_1^2\right]}. \tag{4.5}$$

The *local* equilibrium degree of dissociation is given by

$$(1 - Y_{1_e})e^{-\theta_d/T} - \frac{\rho}{\rho_d}Y_{1_e}^2 = 0 \tag{4.6}$$

The previous two equations can be combined to give

$$\frac{1}{\tau} = \Phi T^\eta \rho \left[e^{-\theta_d/T} + \frac{\rho}{\rho_d}(Y_1 + Y_{1_e})\right]. \tag{4.7}$$

Note that the above expression gives a non-zero value for the characteristic time-scale when $U \longrightarrow U_e$ for finite values of the reaction parameter Φ, whereas Equation (4.5) yields an indeterminate value.

For the purposes of stability analysis, the equilibrium state and the characteristic velocity and-time-scales are regarded as constants and hence the transformation $U \longrightarrow U - U_e$ can be used to simplify the scalar model equation to

$$\frac{\partial U}{\partial t} + u\frac{\partial U}{\partial x} = -\frac{U}{\tau}. \tag{4.8}$$

The flux and source Jacobians of this model are also constants

$$\frac{\partial F}{\partial U} = u \quad , \qquad \frac{\partial W}{\partial U} = -\frac{1}{\tau}. \tag{4.9}$$

The fully implicit methods usually require the inversion of a block multi-diagonal system of algebraic equations. This is more complicated than the source implicit scheme and the realization of full advantage of vector processing machines for adaptive algorithms becomes difficult. Hence only the source implicit and fully explicit algorithms will be examined here. For the Ni scheme the spatial grid will be regarded as uniform ($\epsilon_j \equiv 0$ for all nodes) in the absence of spatial and temporal adaptation and artificial viscosity will not be applied. The cell changes for a structured grid for cells B and C in Figure (3.1) are as follows

$$\begin{aligned} \Delta U^*_{jB} &= (F_{j-1} - F_j)\,\tfrac{\Delta t}{\Delta x} + W^*_j \Delta t &&= \Gamma\,(U_{j-1} - U_j) - DU^*_j \\ \Delta U^*_{jC} &= (F_j - F_{j+1})\,\tfrac{\Delta t}{\Delta x} + W^*_j \Delta t &&= \Gamma\,(U_j - U_{j+1}) - DU^*_j \end{aligned} \tag{4.10}$$

where the terms without a superscript are evaluated explicitly or at a time-level (n), whereas asterisks indicate terms which may be treated implicitly, *i.e.*,

$$U^*_j = \begin{cases} U^n_j & \text{for explicit schemes} \\ U^{n+1}_j & \text{for implicit schemes} \end{cases} \tag{4.11}$$

and the CFL number Γ and grid Damköhler number D (Damköhler number is the ratio of the convection time-scale t_r and the reaction chemical time-scale) are given by

$$\Gamma = u\frac{\Delta t}{\Delta x} \quad , \qquad D = \frac{\Delta t}{\tau}. \tag{4.12}$$

The stiffness of the scalar model increases with the magnitude of the grid Damköhler number. The explicit flux changes and source changes are given by

$$\Delta F_{jP} = u\Delta U_{jP} \quad , \qquad \Delta W^*_{jP} = -\frac{\Delta U^*_{jP}}{\tau} \tag{4.13}$$

First Order	Second Order	Amplification Factor	Bounding Curve
E	E	$1 - D + 0.5D^2 + \Gamma^2(\cos\theta - 1)$ $+0.5\Gamma I \sin\theta(D-2)$	$\Gamma^2 \leq 1 - \frac{D}{2} + \frac{D^2}{4}$, $D \leq 2$
E	I	$\frac{1-D+\Gamma^2(\cos\theta-1)+0.5\Gamma I\sin\theta(D-2)}{1-0.5D^2}$	$\Gamma^2 \leq 1 - \frac{D}{2} - \frac{D^2}{4}$, $D \leq \sqrt{5}-1$
E	N	$1 - D + \Gamma^2(\cos\theta - 1) - \Gamma I \sin\theta$	$\Gamma^2 \leq 1 - \frac{D}{2}$, $D \leq 2$
I	E	$\frac{1+0.5D^2+\Gamma^2(\cos\theta-1)+0.5\Gamma I\sin\theta(D-2)}{1+D}$	$\Gamma^2 \leq 1 + \frac{D}{2} + \frac{D^2}{4}$, $D \leq 2$
I	I	$\frac{1+\Gamma^2(\cos\theta-1)+0.5\Gamma I\sin\theta(D-2)}{1+D-0.5D^2}$	$\Gamma^2 \leq 1 + \frac{D}{2} - \frac{D^2}{4}$, $D \leq 2$
I	N	$\frac{1+\Gamma^2(\cos\theta-1)-\Gamma I\sin\theta}{1+D}$	$\Gamma^2 \leq 1 + \frac{D}{2}$, $D \geq 0$

Table 4.1: Summary of stability regions for 1-D scalar equation, the letters E, I, N respectively stand for explicit, implicit, nil.

where the subscript P denotes either cell B or C. Substituting these equations in Equation (3.22) yield the change contributions at node j

$$\begin{aligned} \delta U_{jB} &= \tfrac{1}{2}\left[\Delta U^*_{jB} + \Gamma \Delta U_{jB} - q\tfrac{D}{2}\Delta U^*_{jB}\right] \\ \delta U_{jC} &= \tfrac{1}{2}\left[\Delta U^*_{jC} - \Gamma \Delta U_{jC} - q\tfrac{D}{2}\Delta U^*_{jC}\right]. \end{aligned} \tag{4.14}$$

Using Equations (4.10) to (4.14) the overall change at node j can now be written as

$$\begin{aligned} \delta U_j \equiv U_j^{n+1} - U_j^n &= \frac{\Gamma}{2}(U_{j-1} - U_{j+1}) + \frac{\Gamma^2}{2}(U_{j-1} - 2U_j + U_{j+1}) \\ &\quad -DU_j^* - \frac{qD}{4}\left\{\Gamma(U_{j-1} - U_{j+1}) - 2DU_j^*\right\}. \end{aligned} \tag{4.15}$$

In this equation the first order source term is $-DU_j^*$ and the second order source term has the factor q. For the stability analysis, let us define the Fourier components

$$\begin{aligned} U_j^n &= G^n e^{I\omega x} \\ U_j^{n+1} &= G^{n+1} e^{I\omega x} \end{aligned} \tag{4.16}$$

where G is the amplification factor, ω is the wave number and I represents square-root of -1. This equation defines the following relations

$$\begin{aligned} U_{j-1} \pm 2U_j + U_{j+1} &= 2U_j(\cos\theta \pm 1) \\ U_{j+1} - U_{j-1} &= 2IU_j \sin\theta \end{aligned} \tag{4.17}$$

where $\theta = \omega\Delta x$ is the phase angle, Equation (4.15) yields the amplification factor

$$G = 1 + \Gamma^2(\cos\theta - 1) - \Gamma I \sin\theta + \frac{q}{2}D\Gamma I \sin\theta - DG^\beta + \frac{q}{2}D^2G^\beta \tag{4.18}$$

where

$$G^\beta = \begin{cases} 1 & \text{for explicit schemes} \\ G & \text{for implicit schemes.} \end{cases} \tag{4.19}$$

A scheme is stable if the magnitude of the amplification factor remains less than unity. Table (4.1) shows the various schemes which Equation (4.18) represents and the corresponding bounding curves for the stable regions. The letters E, I and N stand for explicit, implicit and nil respectively. These schemes are presented here to establish a basis for the best possibilities, on which future developments will proceed.

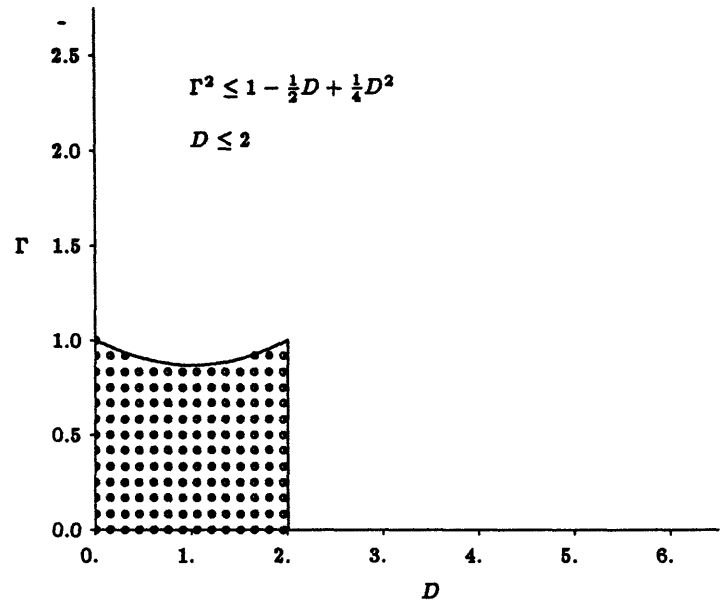

Figure 4.1: Stability curve for explicit first and second order source terms (EE).

The stability regions for various schemes in Equation (4.18) are determined on a suitable grid in the Γ-D plane. For the figures shown here this grid spans the region $\Gamma \in [0,2.5]$, $D \in [0,6]$ with 41 points along D-axis and 31 points along Γ-axis. The phase angle θ was varied from 0 to π radian in equal increments ($\pi/36$ radian) and the norm of the amplification factor was checked at each node of the Γ-D plane. The nodes for which $|G| \leq 1$, for every discrete value of $\theta \in [0,\pi]$, were marked by a small circle to indicate those that belong to the stable region of the scheme. Computations on a finer grid yield essentially the same stability regions, *i.e.*, the finer grid merely involves more dots and yields the same bounding curves. This procedure has the disadvantage that information about individual Fourier components is lost; however, the interest here was solely to determine the stability regions.

The lower bounds for the region of stability in all cases are obviously $D \geq 0$ and $\Gamma \geq 0$. The upper bounds for most of the cases are dictated by the $\theta = \pi$ Fourier component. Figure (4.1) shows the stability domain for the EE scheme when both the first and second order source terms are explicit. The upper bound on the grid Damköhler number is 2, which can be a severe limitation when the chemical time-scale

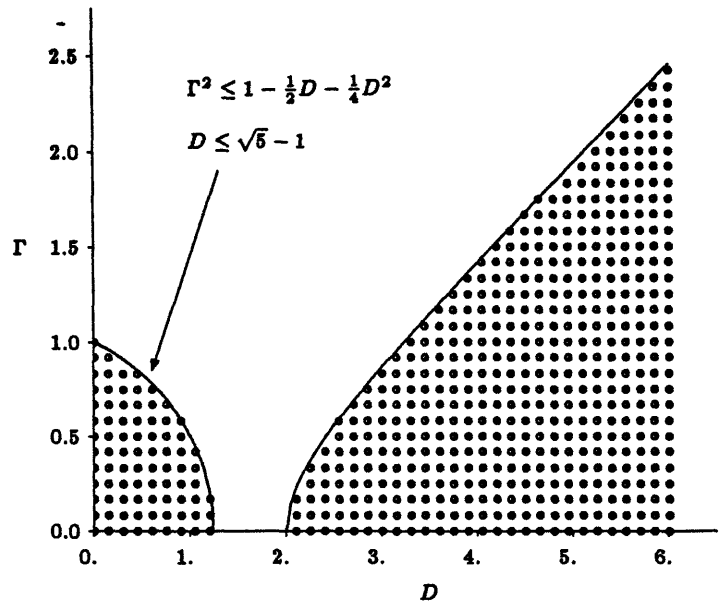

Figure 4.2: Stability curve for explicit first order and implicit second order source term (EI).

is small.

Figure (4.2) shows the stability domain for the EI scheme, *i.e.*, when the first order source term is modelled explicitly and the second order source term implicitly. A doubly connected stable region is apparent in this figure. A numerical experiment with cases involving disjoint regions in which $|G| \leq 1$ shows that the schemes are stable and monotone only in the region which contains the origin $\Gamma = D = 0$ which is the limit point for $\Delta t \rightarrow 0$ with Δx fixed. Alternately, if the time-step is gradually increased in a numerical experiment, the scheme will become unstable when D first becomes more than $\sqrt{5}-1$ and the experiment would be aborted before the time-step has the chance to achieve D values greater than 2; thereby making the stability in this region immaterial. The implication that the stable domain corresponds to the simply connected region containing the arbitrarily small cell dimensions is possibly of a general type, although a strict proof may be difficult. Thus, it is noted that instead of a gain in the stability, compared to the EE scheme, the EI scheme is much more restrictive.

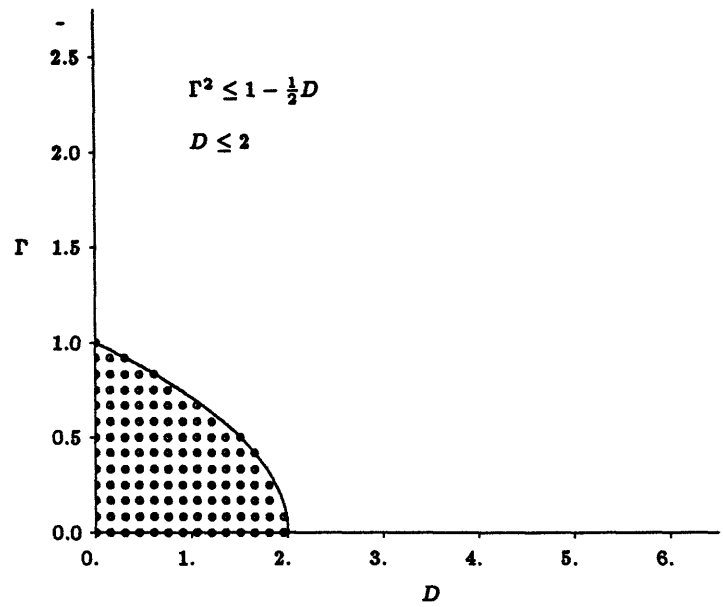

Figure 4.3: Stability curve when first order source term is explicit and second order source term is excluded (EN).

Figure (4.3) shows the stability domain for the EN scheme in which the first order source term is explicit and the second order source term is not retained. As expected the stability region is more restricted compared to the EE scheme; however it is slightly better compared to the EI scheme.

The stable region for the IE scheme, in which the dominant source term is implicit and the next order terms are explicit, is presented in Figure (4.4). The stability region is enhanced compared to all other schemes in which the dominant source term was explicit. However, the stability is still restricted by the $D \leq 2$ constraint. This is a manifestation of the quadratic term in D; as the time-step increases the explicit quadratic term becomes more dominant compared to the linear implicit term.

The fully implicit or II scheme stability curve is shown in Figure (4.5). Again the amplitude limiting region is composed of two distinct regions; however the stable and monotone region is the one which contains the origin. The stability region is somewhat

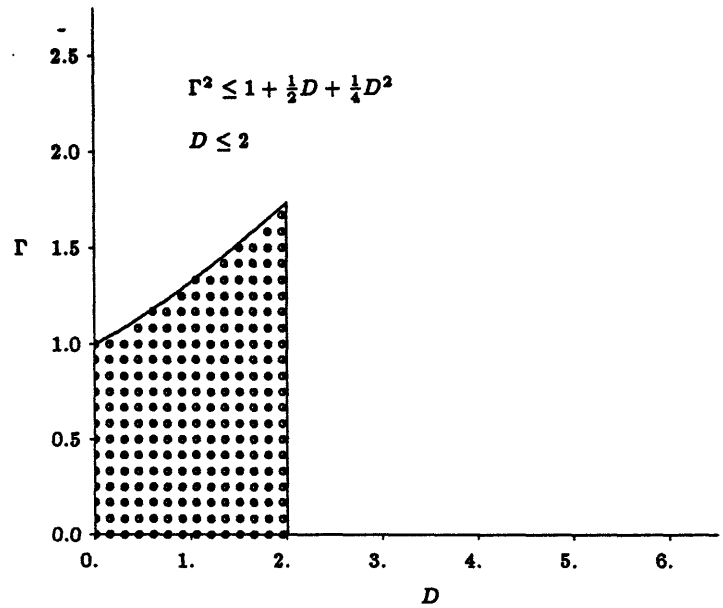

Figure 4.4: Stability curve for implicit first order and explicit second order source term (IE).

enhanced compared to the EE scheme although its performance is slightly worse compared to the IE scheme. The stability is still restricted by the $D \leq 2$ constraint. A numerical experiment with this scheme, when the convective term is set zero, indicates that for $D > 1 + \sqrt{5}$, the sign of U_j^{n+1} becomes reverse of the sign of U_j^n, although the norm of the amplitude factor does not exceed unity; *i.e.*, the value of the dependent variable oscillates about zero with a slowly diminishing amplitude. Hence for problems in which *positivity* ($U_j^n, U_j^{n+1} \geq 0$) is important, the stable region should not only limit the norm of the amplitude function but also preserve the positivity condition. For all the schemes involving disjoint regions of $|G| \leq 1$, the region containing the arbitrarily small time-steps is the only one that preserves positivity.

As evident from Figure (4.6) for the IN scheme, in which only the first order source term is retained implicitly, the stability becomes independent of grid Damköhler number and is only constrained by the CFL restriction $\Gamma \leq \sqrt{1 + D/2}$. This is due to the fact that the quadratic term in D has been excluded and the implicit first order source term

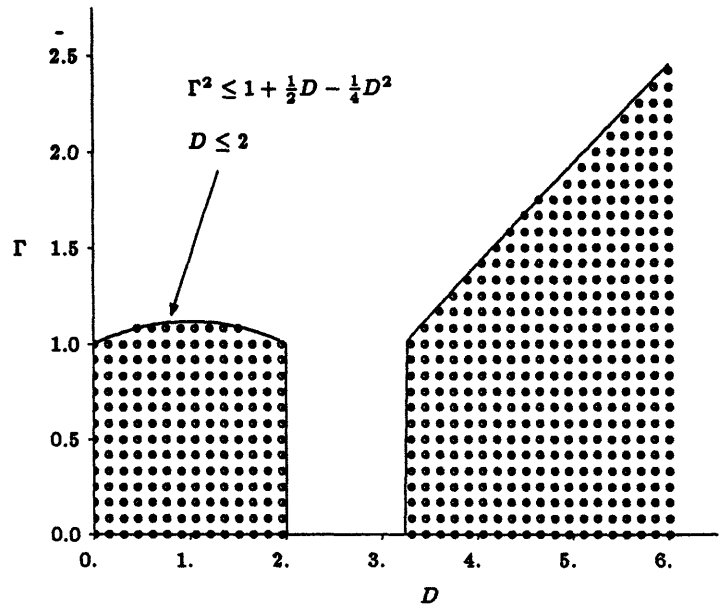

Figure 4.5: Stability curve for implicit first and second order source terms (II).

provides a preconditioning (multiplication by $1/(D+1)$), the effect of which increases with the increasing time-step. If the sign of the second order real source term ($\frac{1}{2}D^2G$) had been the same as the first order source term (DG) then the II scheme would have had better stability characteristics than the IN scheme. This is consistent with the findings of [23,24,42,122] who have used only the first order implicit source terms. For all of the other schemes there is no substantial gain in stability over the fully explicit scheme. The principal advantage of the IN scheme is that the numerical time-step becomes independent of the chemical time-scales. The disadvantages include the fact that the scheme is less accurate at small time-steps and it is computationally more complicated compared to the EE scheme. It would be misleading to conclude that, depending upon the value of grid Damköhler number, CFL numbers greater than unity can be selected. It is worthwhile to remember that the analysis only holds for a single scalar equation and not for a system of equations. There are other scalar equations, *e.g.* global continuity equation, where the source term is zero and the correct limit on CFL number is then unity. In the following section the behavior of both the fully explicit and IN schemes is examined on a model problem.

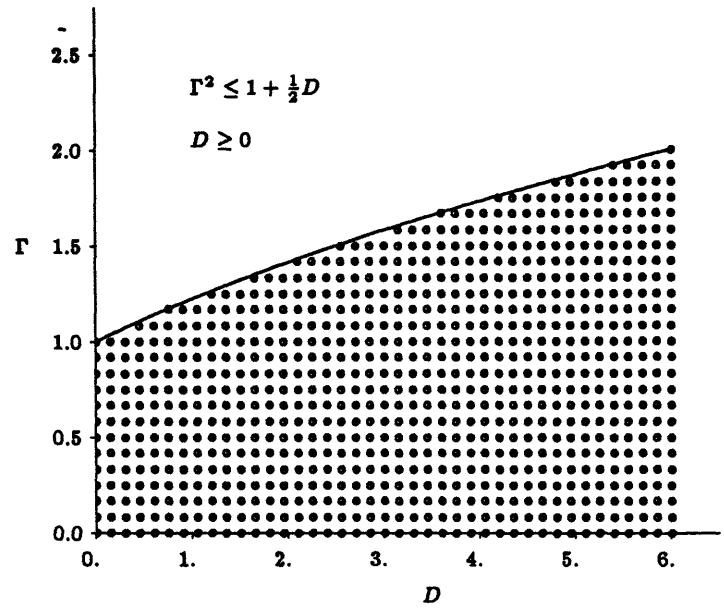

Figure 4.6: Stability curve when first order source term is implicit and second order source term is excluded (IN).

It is interesting to note that Equation (4.8) has an exact solution, for a single wave number ω, which describes the temporal decay of a periodic sinusoidal profile. This solution can be determined by the *separation of variables* technique and has the form

$$U(x,t) = Ae^{-t/\tau}e^{I\omega(x-ut)} \tag{4.20}$$

where A is the initial amplitude of the periodic profile. The solution indicates that the amplitude decreases monotonically with time and asymptotically approaches zero. The phase of the harmonic associated with the profile shifts as function of time while the frequency remains the same (the phase shifts by ωut to the right after time t). The amplitude function of the exact solution is

$$G = e^{-D}\left[\cos(\Gamma\theta) - I\sin(\Gamma\theta)\right]. \tag{4.21}$$

The norm of this amplitude function is always less than unity, since $D \geq 0$; hence the physical situation always represents a stable system although the numerical schemes may be subject to instabilities.

The ratio ϕ_r of the phase shift of the IN scheme in the exact solution is

$$\phi_r = \frac{1}{\Gamma\theta}\sin^{-1}\left(\frac{\Gamma\sin\theta}{1+D}\right). \tag{4.22}$$

This ratio is $1/(1+D)$ at $\theta = 0$ and 0 at $\theta = \pi$ radian.

The variations of both the amplitude and phase shift with the wave number show that the IN scheme behaves reasonably well at high frequencies (small θ) and is more accurate than at low frequencies. However, the low frequency parts of the solution, where numerical errors are worst, decay rapidly and the solution becomes smoother as time progresses.

4.4 Exact Solution of a Localized 1-D Source Model

Consider integrating the simplest case, *i.e.*, when the convective term is neglected ($\Gamma = 0$)

$$\frac{\partial U}{\partial t} = -\frac{U}{\tau}. \tag{4.23}$$

This model has an analytic solution which can be compared to the numerical schemes to assess the temporal order of accuracy. The exact solution is

$$U_j^n = U_j^0\ \mathrm{e}^{-t/\tau} \tag{4.24}$$

where U_j^0 is the initial value of the dependent variable at time $t = 0$ and node j. The solution decays exponentially from its initial value to zero (or U_e in terms of the original variable), which is the equilibrium value for this case. The exact solution can be written in the delta form as

$$\delta U_j = U_j^0\ \mathrm{e}^{-(t+\Delta t)/\tau} - U_j^0\ \mathrm{e}^{-t/\tau} = U_j^n\ (\mathrm{e}^{-\Delta t/\tau} - 1). \tag{4.25}$$

The Taylor's series expansion, about zero, of this delta form is

$$\delta U_j = U_j^n\ (-\frac{\Delta t}{\tau} + \frac{\Delta t^2}{2\tau^2} - \frac{\Delta t^3}{6\tau^3} + \cdots). \tag{4.26}$$

The numerical solution of Equation (4.23) is given by Equation (4.15) with $\Gamma = 0$, *i.e.*,

$$\delta U_j = U_j^*(-\frac{\Delta t}{\tau} + \frac{q}{2}\frac{\Delta t^2}{\tau^2}). \tag{4.27}$$

Obviously the fully explicit EE algorithm matches the exact solution for $\Delta t/\tau \ll 1$ to second order and hence it is second order accurate in time. The implicit scheme of interest; *viz.* the IN scheme, yields the value

$$U_j^{n+1} = \frac{U_j^n}{1+\frac{\Delta t}{\tau}} \tag{4.28}$$

or in delta form

$$\delta U_j = -U_j^n \frac{\Delta t}{\tau} \frac{1}{1+\frac{\Delta t}{\tau}}. \tag{4.29}$$

The Taylor's series expansion of this solution is

$$\delta U_j = U_j^n(-\frac{\Delta t}{\tau} + \frac{\Delta t^2}{\tau^2} - \frac{\Delta t^3}{\tau^3} + \cdots). \tag{4.30}$$

The expansion does have a second order term but the solution is exact only to first order; hence the scheme is temporally first order accurate for small time-steps. It can be observed that a hybrid algorithm that sets $q = 0$ but regards U_j^* to be the average of U_j^n and U_j^{n+1} would yield a second order accurate solution for small time-steps. This scheme will be referred to as the CN (Crank-Nicolson) scheme. The delta form for this hybrid scheme is

$$\delta U_j = -\frac{1}{2}(U_j^n + U_j^{n+1})\frac{\Delta t}{\tau} \tag{4.31}$$

which yields

$$U_j^{n+1} = \frac{1-\frac{\Delta t}{2\tau}}{1+\frac{\Delta t}{2\tau}} U_j^n \tag{4.32}$$

and hence

$$\delta U_j = -U_j^n \frac{\Delta t}{\tau} \frac{1}{1+\frac{\Delta t}{2\tau}} = U_j^n(-\frac{\Delta t}{\tau} + \frac{\Delta t^2}{2\tau^2} - \frac{\Delta t^3}{4\tau^3} + \cdots). \tag{4.33}$$

This is second order accurate and yet has an implicit source term component. The stability domain of this scheme is restricted by $0 \leq \Gamma \leq 1$ and is independent of the chemical time-scales. These properties are attractive; however, as will be shown later, this scheme does not satisfy the *positivity* condition like the IN scheme. Positivity means that quantities such as species mass fraction, cannot become negative during the course of integration.

Figures (4.7) to (4.9) show the numerical solution for three choices of time-steps, *i.e.*, $\Delta t = \tau, 2\tau, 3\tau$, along with the exact solution. The line-segments marked with symbols correspond to fully explicit EE, source implicit IN and hybrid CN schemes and the

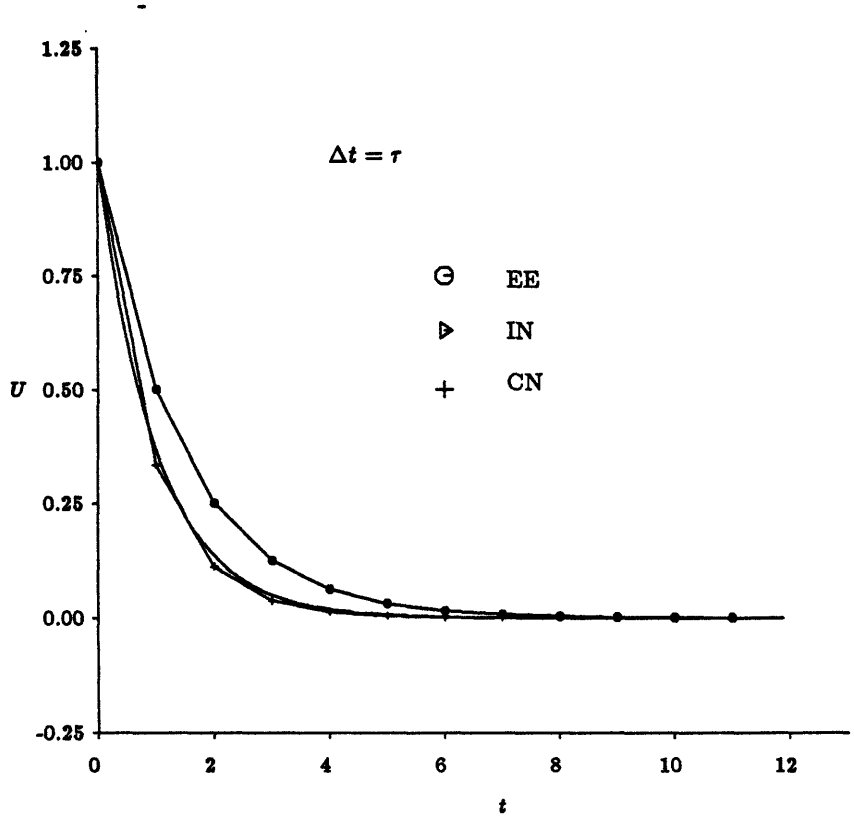

Figure 4.7: Solutions for a localized scalar model with $\Delta t = \tau$.

exact solution is an unmarked curve. For the $\Delta t = \tau$ case all of the numerical solutions replicate the exact solution and are numerically stable. The same is true for other schemes in Table (4.1); although these schemes are not shown in Figure (4.7). The EE scheme is nearly indistinguishable from the IN scheme and their decay is relatively slow compared to the true solution. The description of the CN solution is very close to the exact solution and falls below it.

When the time-step is $\Delta t = 2\tau$, the changes computed by the EE scheme are zero, the solution remains at the initial condition of $U = 1$ and does not exhibit the decaying process. This situation marks the borderline of classical mathematical stability for the explicit schemes. The CN scheme goes exactly to zero in one time-step and stays at

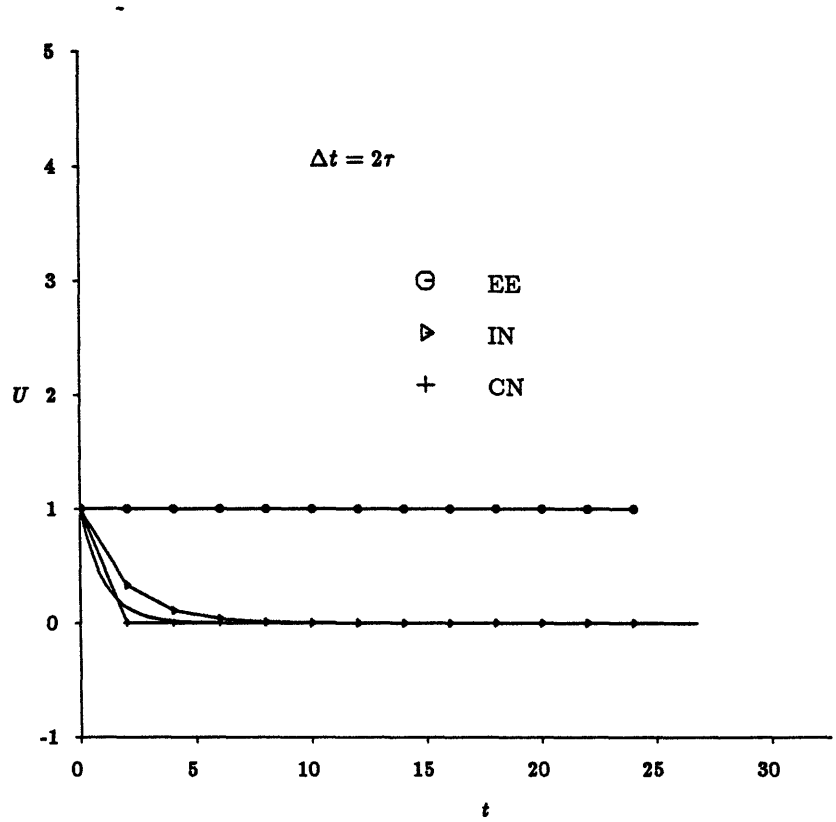

Figure 4.8: Solutions for a localized scalar model with $\Delta t = 2\tau$.

that equilibrium state thereafter. The asymptotic limit is achieved far too early. The IN scheme exhibits the relaxation process qualitatively and can be regarded superior to both EE and CN schemes for this value of the time-step.

For $\Delta t = 3\tau$ the changes computed by the EE scheme successively increase and eventually the solution becomes unstable. The CN scheme exhibits lack of positivity when the solution becomes negative after the first time-step. However, the solution recovers and approaches the true asymptotic behavior. The IN scheme preserves the positivity condition and tends to the correct equilibrium limit, although the decay process lags behind the true solution for the first few time-steps. Care must be exercised in choosing small time-steps at the initial stage of a relaxation process so that the transient is cap-

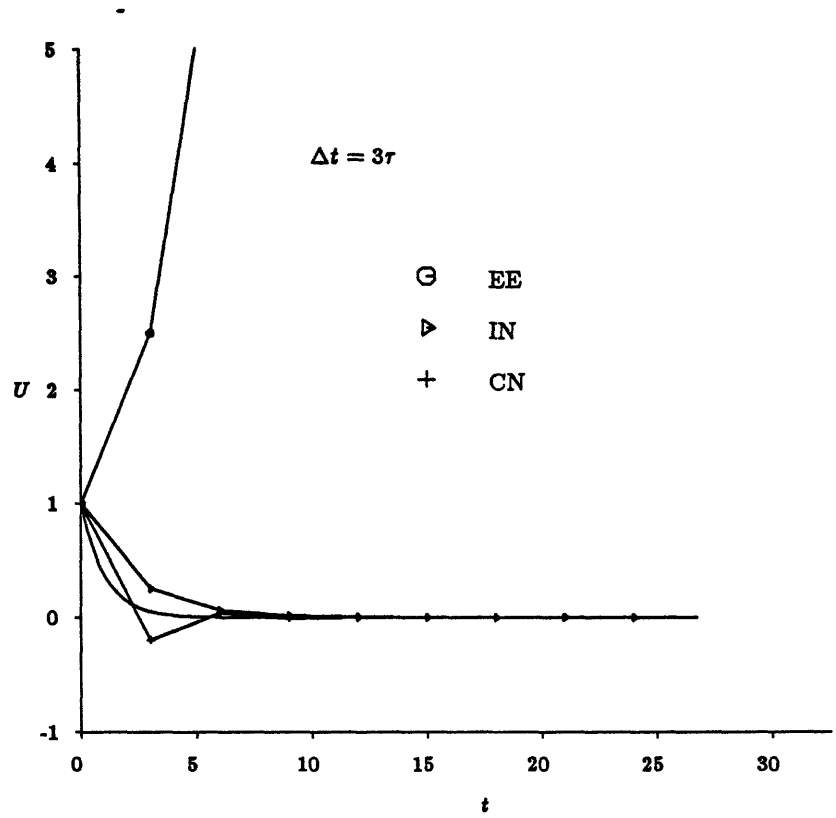

Figure 4.9: Solutions for a localized scalar model with $\Delta t = 3\tau$.

tured correctly; however, once the initial transient is completed larger time-steps may be selected.

It can be concluded that among all the schemes examined here the IN scheme is the most cost-effective scheme for stiff reaction systems, although it is not as accurate as the EE or CN schemes for small time-steps. It is also appropriate to point out that the IN scheme discussed here does not modify the transient history if the preconditioning matrix has the form as described here and the Jacobian terms are evaluated correctly. The only restriction is for the temporal order of accuracy and hence smaller time-steps should be selected in the regions where large temporal gradients are expected.

4.5 Implementation of Source Implicit Scheme

An approach for treating the source terms implicitly has been discussed in Chapter 3, which uses the preconditioning matrix on the cell changes. An alternate approach which utilizes the preconditioning matrix on the distribution formulae is discussed here. This approach is detailed for one spatial dimension and is then generalized for the 2-D case. The implicit source term for node j in Figure (3.1) can be expanded by Newton linearization

$$W_j^{n+1} = W_j^n + \left.\frac{\partial W}{\partial U}\right|_j (U_j^{n+1} - U_j^n) = W_j^n + \left.\frac{\partial W}{\partial U}\right|_j (\delta U_{jB} + \delta U_{jC}). \tag{4.34}$$

Hence the implicit cell change for cell B is given by

$$\Delta U_{jB}^{IM} = \frac{\Delta t_B}{\Delta x_B}(F_i - F_j) + W_j^n \Delta t_B + \left.\frac{\partial W}{\partial U}\right|_j (\delta U_{jB} + \delta U_{jC})\Delta t_B \tag{4.35}$$

or in terms of the explicit cell change

$$\Delta U_{jB}^{IM} = \Delta U_{jB}^{EX} + \left.\frac{\partial W}{\partial U}\right|_j (\delta U_{jB} + \delta U_{jC})\Delta t_B. \tag{4.36}$$

It is assumed here that $\Delta t_B = \Delta t_C$; a similar expression for cell C is

$$\Delta U_{jC}^{IM} = \Delta U_{jC}^{EX} + \left.\frac{\partial W}{\partial U}\right|_j (\delta U_{jB} + \delta U_{jC})\Delta t_C. \tag{4.37}$$

These expressions are used in conjunction with the distribution formulae, Equations (3.22). The source change contributions for these distributions are zero since here $q = 0$ and the flux changes remain the same since the cell changes for these remain explicit. Substituting the implicit cell changes in the distribution formulae and summing the individual contributions at node j yields

$$\begin{aligned}
\delta U_{jB} + \delta U_{jC} \;=\; & \frac{1-\epsilon_j}{2}\left[\Delta U_{jB}^{EX} + \frac{\Delta t_B}{\Delta x_B}\left(\Delta F_B + \frac{2\epsilon_j}{1-\epsilon_j}\Delta F_{jB}\right)\right] + \\
& \frac{1+\epsilon_j}{2}\left[\Delta U_{jC}^{EX} - \frac{\Delta t_C}{\Delta x_C}\left(\Delta F_C - \frac{2\epsilon_j}{1+\epsilon_j}\Delta F_{jC}\right)\right] + \\
& \left.\frac{\partial W}{\partial U}\right|_j (\delta U_{jB} + \delta U_{jC})\Delta t_C.
\end{aligned}$$

Decomposing this back into contributions from cells B and C yields

$$\begin{aligned}
\left(I - \left.\tfrac{\partial W}{\partial U}\right|_j \Delta t_B\right)\delta U_{jB}^{IM} \;=\; \delta U_{jB}^{EX} \;\equiv\; & \tfrac{1-\epsilon_j}{2}\left[\Delta U_{jB}^{EX} + \tfrac{\Delta t_B}{\Delta x_B}\left(\Delta F_B + \tfrac{2\epsilon_j}{1-\epsilon_j}\Delta F_{jB}\right)\right] \\
\left(I - \left.\tfrac{\partial W}{\partial U}\right|_j \Delta t_C\right)\delta U_{jC}^{IM} \;=\; \delta U_{jC}^{EX} \;\equiv\; & \tfrac{1+\epsilon_j}{2}\left[\Delta U_{jC}^{EX} - \tfrac{\Delta t_C}{\Delta x_C}\left(\Delta F_C - \tfrac{2\epsilon_j}{1+\epsilon_j}\Delta F_{jC}\right)\right].
\end{aligned} \tag{4.38}$$

Now that the individual contributions are derived, different time-steps for cells B and C can be allowed. The overall distribution formulae for cell C including artificial viscosity are given by Equations (3.35) which generalize to

$$\delta U_{jC} = \frac{1+\epsilon_j}{2}\left[I-(1-q)\left.\frac{\partial W}{\partial U}\right|_j \Delta t_C\right]^{-1}\left[\Delta U_j - \frac{\Delta t}{\Delta x}\left(\Delta F - \frac{2\epsilon_j}{1+\epsilon_j}\Delta F_j + \Psi\right) + q\frac{\Delta t}{2}\Delta W_j\right]_C$$
$$\delta U_{kC} = \frac{1-\epsilon_k}{2}\left[I-(1-q)\left.\frac{\partial W}{\partial U}\right|_j \Delta t_C\right]^{-1}\left[\Delta U_k + \frac{\Delta t}{\Delta x}\left(\Delta F + \frac{2\epsilon_k}{1-\epsilon_k}\Delta F_k + \Psi\right) + q\frac{\Delta t}{2}\Delta W_k\right]_C . \tag{4.39}$$

These now hold for both fully explicit ($q = 1$) and source implicit ($q = 0$) schemes. Note that in these relations the source term Jacobians are evaluated at the nodes and the cell changes, as given by Equation (3.13), also involve the source terms at the nodes.

For the 2-D case all Jacobians in the STAR code are evaluated at the cell centers in the spirit of a finite volume approach. The generalization of the distribution formulae of Equations (3.70) is now straight-forward; as an example the contribution to node i is

$$\begin{aligned}\delta U_{iC} \;=\; & \frac{1}{4}\left[I-(1-q)\left.\frac{\partial W}{\partial U}\right|_C \Delta t_C\right]^{-1} \\ & \left[\Delta U - \frac{\Delta t}{A}\left(\Delta y_{ns}\Delta F - \Delta x_{ns}\Delta G\right) - \frac{\Delta t}{A}\left(\Delta x_{ew}\Delta G - \Delta y_{ew}\Delta F\right) + q\frac{\Delta t}{2}\Delta W + \Psi_i\right]_C .\end{aligned} \tag{4.40}$$

Note that when the integration is carried out on a cell by cell basis the preconditioning matrix need be inverted only once per integration.

4.6 Modification of Source Vector

Consider the variation of species density in the absence of convective term, a first order integration of the species equation gives

$$(\rho Y_s)^{n+1} = (\rho Y_s)^n + \Delta t W_s^n . \tag{4.41}$$

The species whose density is in greatest danger of being driven negative is the one for which $(\rho Y_s)^n$ is small and W_s^n is a large negative number. Such a species will be referred to as *nenspec* which is the acronym for Negatively ENdangered SPECies. Recall that the overall source term for species s may have contributions from all reactions and hence

there may be one nenspec for each reaction. A suitable criterion for justifying that the species k in reaction r is nenspec is the following

$$\hat{m}_k(\beta_{kr} - \alpha_{kr})(\Omega_{fr} - \Omega_{br})/\rho Y_k^n < \Lambda_{min} < 0 \tag{4.42}$$

where Λ_{min} is a pre-selected non-dimensional negative value. The expression on the left side must be minimum for nenspec among all the species which take part in the reaction. The requirement that the above expression be strictly negative debars the inert species ($\beta_{sr} = \alpha_{sr}$) from being a nenspec candidate. A given reaction may not have a nenspec associated with it, in which case the possibility that the reaction causes any of the species densities to go negative is remote. If all of the reactions are devoid of nenspec then the time-step Δt may not have to be restricted beyond the CFL constraint. On the other hand if any of the reactions has a nenspec then the time-step may have to be reduced, often prohibitively, or implicit schemes may have to be used. In what follows an alternative cure is proposed to counter this behavior.

It is obvious that $(\rho Y_s)^{n+1}$ can not be negative physically. For explicit schemes the situation is controlled by taking extremely small time-steps, so that the product $\Delta t\, W_s^n$ is a small negative number and its sum with $(\rho Y_s)^n$ yields either a smaller species density or at most zero. For implicit schemes the source term is replaced by W_s^{n+1} and if this scheme yields appropriate results, *i.e.*, results in smaller final value of species density, then its effect is

$$|W_s^{n+1}| < |W_s^n|.$$

If this condition is not met, *i.e.*, if the implicit source term is as large a negative number as the explicit source term, the species density will be driven negative even for implicit schemes and special reapportionment of species density will have to be carried out to preserve positivity [5]. Thus it is reasonable to choke or reduce the value of W_s^n for the reaction for which a nenspec exists and if explicit terms are desired. Consider the numerator of the expression in Inequality (4.42) in expanded form

$$\hat{m}_k \left\{(\beta_{kr}\Omega_{fr} + \alpha_{kr}\Omega_{br}) - (\beta_{kr}\Omega_{br} + \alpha_{kr}\Omega_{fr})\right\}.$$

Note that the stoichiometric coefficients and the progress rates are all positive numbers and hence the parenthetical quantities in the above expression are positive. In an

extreme case when only the large negative terms of the k^{th} species are important, the contribution of reaction r to the source term of species k is approximately

$$W_{min,r} \approx -\hat{m}_k(\beta_{kr}\Omega_{br} + \alpha_{kr}\Omega_{fr}). \tag{4.43}$$

The subscript r again emphasizes that there may be one such quantity for each reaction. The contribution to $(\rho Y_s)^{n+1}$ from this extreme reaction (when the contributions from other reactions are small) is

$$(\rho Y_k)^n \approx (\rho Y_k)^{n+1} + \Delta t\hat{m}_k(\beta_{kr}\Omega_{br} + \alpha_{kr}\Omega_{fr}). \tag{4.44}$$

The *choking factor* for this reaction must then be based on species k and a suitable form for it is

$$CF_r = \frac{(\rho Y_k)^n}{(\rho Y_k)^n + c\Delta t\hat{m}_k(\beta_{kr}\Omega_{br} + \alpha_{kr}\Omega_{fr})} \tag{4.45}$$

The constant c has values 1 or 0 depending upon whether the nenspec k for the reaction r exists or not. The second factor in the denominator is zero when nenspec does not exist and, depending upon the strength of the nenspec, it could be a very small positive number. The time-step is set equal to that of a cell which is being integrated when the solver is applied. The general form of the modified source term is then

$$W_s = \hat{m}_s(\beta_{sr} - \alpha_{sr})(\Omega_{fr} - \Omega_{br})CF_r. \tag{4.46}$$

For the case when $\Delta t \to 0$, the explicit and modified source terms are essentially the same, whereas for the case $\Delta t \to \infty$, the modified source term approaches zero and hence there is no danger of divergence of the mass fractions in the negative sense.

The choking factor in the previous analysis has been obtained in an *ad-hoc* manner. In the following the exact form of this factor will be justified. Consider that $U^n = (\rho Y_s)^n$ is a small positive number and W_s^n is a large negative number which can be approximated by $W_{min,r}$ of Equation (4.43) and that only one reaction is the dominant one. The partial differential equation to be solved is

$$\frac{\partial U}{\partial t} = W^{n+1} = W^n + \frac{\partial W}{\partial U}\frac{\partial U}{\partial t}\Delta t. \tag{4.47}$$

For the sake of computing the source Jacobians assume that

$$U^{n+1} \to 0, \qquad W^{n+1} \to 0$$

hence

$$\frac{\partial W}{\partial U} = \frac{W^{n+1} - W^n}{U^{n+1} - U^n} \approx -\frac{\hat{m}_k}{U^n}(\beta_{kr}\Omega_{br} + \alpha_{kr}\Omega_{fr}). \tag{4.48}$$

Substituting this in Equation (4.47) yields

$$\frac{\partial U}{\partial t} = \frac{\rho Y_k}{\rho Y_k + \Delta t \hat{m}_k(\beta_{kr}\Omega_{br} + \alpha_{kr}\Omega_{fr})} W^n. \tag{4.49}$$

The factor multiplying the explicit source term is the choking factor for one reaction which has modified the source term and has the same form as Equation (4.45).

Another problem with chemical source terms occurs when $(\rho Y_s)^n$ is large (Y_s approaches its maximum possible value $Y_{max,s}$) and W_s is a large positive number, then the species mass fraction is in danger of increasing beyond its maximum possible value. An analysis for this dangerous situation is unnecessary since for $Y_k > Y_{max,k}$ there is a species l for which $Y_l < 0$, and this case has already been discussed.

As noted earlier, the prescription described here is tantamount to making the part of $\partial \rho Y_s / \partial t$ that is due to reaction r linearly implicit in ρY_s which prevents the mass fraction from being driven negative for large values of time-steps. Approaches similar to the one described here are presented in References [5,114]. Unlike the IN scheme discussed in the previous sections the inversion of a preconditioning matrix is not needed for this approach. Another advantage is that the Jacobian matrices $\partial W / \partial U$ are not really involved in the solution algorithm for the first order schemes ($q = 0$), the computations of these Jacobians can be very expensive especially when large number of reactions are involved. The disadvantage of the approach is that nenspec has to be determined for all reactions which may be computationally expensive for a large number of reactions.

For the model problem discussed in Section (4.4) consider the reaction $A \longrightarrow B$, here the source term for species A can be written as

$$W_A = -\hat{m}_A \left[k_f \frac{\rho Y_A}{\hat{m}_A} - k_b \frac{\rho Y_B}{\hat{m}_B} \right]. \tag{4.50}$$

Since the model problem deals with an irreversible reaction the backward rate coefficient is zero; furthermore the forward rate coefficient can be regarded as

$$k_f = \frac{1}{\tau}. \tag{4.51}$$

Substituting these values in Equation (4.45) yields the following value for the choking factor

$$CF_1 = \frac{1}{1+\frac{\Delta t}{\tau}} \tag{4.52}$$

and in the delta form this yields

$$\delta U_j = -U_j^n \frac{\Delta t}{\tau} \frac{1}{1+\frac{\Delta t}{\tau}} \tag{4.53}$$

which is the same as the solution for the IN scheme. Hence for multiple reactions this approach can be expected to yield consistent results. For most of the cases discussed in this thesis the results obtained by this approach and the IN scheme are essentially the same.

Chapter 5

Spatial Adaptation

This chapter begins by introducing various spatial adaptation techniques and emphasizes embedded mesh concepts. This is followed by a brief introduction of the data-structure utilized for the algorithm. A detailed description of data-structure appears in Appendix C. A multi-variable approach is detailed for unbiased first differences of criteria variables and their threshold values, which are useful in the detection of flow features. Sections (5.4) and (5.5) discuss the grid division and fusion procedures. Section (5.6) details the procedure for enlargement of the spatially resolved region. The chapter concludes by remarking on the avoidance of grid knottiness and a discussion of a block grid generator.

5.1 Motivation

It is well-known that greater accuracy is realized when finer grids are utilized in both space and time. This is because the truncation error of the numerical schemes is dependent upon fineness of the cells; with increasingly finer cells this error tends towards zero. For those limiting conditions the solution of a consistent finite difference analog approaches the exact solution, assuming of course that the round-off error remains negligible, as the cells are refined. It is also well-established that an accurate description of small structures in a flow can be realized generally by spanning the structure with a minimum of three or four computational cells. More cells may be needed accomplish the capture of the feature if steep gradients are involved. The uncertainty pertaining to the location of a particular feature within a cell of course could be reduced by increasing spatial resolution. If the flow structures are not adequately resolved, they become

numerically diffused since a discrete model inherently spreads flow discontinuities over several cells and thereby degrades accuracy. Hence spatial resolution is essential near features like shocks, relaxation zones, vortices, slip lines, *etc.*

It is clear from the CFL constraint that the resolution requirements in space generally imply a corresponding imposition on resolution in time. For most frozen flows this is the primary constraint, but for reacting flows other temporal resolution requirements may be even more stringent than those implied by the spatial resolution. Hence the resolution in time may be controlled only in part by the resolution in space. For cases where strong coupling does exist between the two, allocation of temporal resolution simply follows from that of spatial resolution. For those cases, in one spatial dimension, increasing the spatial resolution by a factor of two imposes a corresponding factor of two in time-steps; hence there is a fourfold increase in computational work to advance to a given interval of time. Similarly, doubling the spatial resolution in two-dimensional flows generally causes the time-steps to reduce to half their previous values which implies an eight-fold increase in computational effort.

The classical way to provide adequate resolution for the capture of features is to use globally fine grids. This usually results in a colossal number of cells which places extensive demands on the CPU memory. This may occasionally exceed the available CPU memory size; although this is not a handicap for a virtual machine, frequent loading and unloading of pages may seriously impair the efficiency of the calculations. Furthermore, as implied earlier, global refinement can result in prohibitively long computational runs. The advantage of a global approach is that the logic is not complicated by a need to manipulate nodes, and a simple *structured* grid suffices. This also reduces the human costs in the sense that changes in the code can be incorporated easily. However, due to the tremendous costs associated with the execution of such programs, the global approach is not a very attractive option. The loss of efficiency can be countered by the use of adaptive techniques, such as moving mesh, zonal approach or local embedding.

In a zonal approach, an overall region is subdivided into zones, and grids within each zone are generated independently according to the desired resolution. This makes

the grid generation process for complicated topologies a simpler task. However, the approach generally results in non-physical boundaries within the overall region due to patched or overlaid grids. The zonal boundaries at the interfaces of various zones must be treated in a special way to ensure conservation. Some typical citations for the zonal techniques are References [8,14,17,63,64,110,113,123].

A second adaptive approach involves redistributing and/or clustering grids in the vicinity of known features. This approach is frequently known as the moving mesh technique. It is generally advocated that numerical methods based on this approach maximize accuracy with a minimum number of grid points. Node movement "functions" are generally defined from the geometry, and propagate nodes into regions having significant discretization errors. However, clustering of cells is very effective when the location of the feature is known *apriori*, at least to some extent, and this clearly is not always the case for unsteady situations. The technique can also introduce substantial cell distortion and an undesirable phenomenon of ***node-entanglement.*** As an example consider the resolution of a feature which revolves around a second feature, via quadrilateral cells, as time progresses. After one complete revolution the nodes should coincide with their initial locations, but generally the distortions gradually increase and the cells are unable to maintain quadrilateral topologies; cell centers may be displaced outside of the cell boundaries, and grid lines may intersect. Such behavior can cause significant errors in computed solutions even for less extreme examples. When grid clustering is used with a global mesh to resolve certain features, clustering also takes place in far field regions resulting in a large number of unnecessary cells there. However, the concept of moderate grid motion coupled with local embedding does present an attractive option for problems in which the domain boundaries are themselves moving. The popularity of moving mesh techniques may be attributed to its relatively straight-forward logic and the structured nature of the grids, although manipulations involving node movements can be somewhat complicated. The technique can be retrofit into an existing structured program with a modest effort. Some of the typical studies, among numerous grid point redistribution schemes, are References [7,47,65,66,91,95,129,130].

Another adaptive approach is local mesh enrichment, in which cells are locally di-

vided to yield additional resolution. Such adaptive embedding algorithms have the advantage that meshes are refined only where necessary and as the solution evolves, thereby providing accurate and relatively inexpensive solutions. Some typical studies in this class are References [10,13,35,99,104,105,119,128]. Those which couple multiple-grids [33,87,133] with locally embedded grids have some aspects in common with the zonal approach. This is because the grids of different coarseness levels are not assembled into a global grid but are stored independently, and different approaches are applied at different levels. For these multiple-grid algorithms the fine grid boundaries overlap the coarse grid boundaries; however unlike the zonal approach, the multiple-grid embedded mesh approach can dynamically change the grid structure as the solution evolves. Since the local embedding can be carried out in a recursive manner, very fine grid spacing can be maintained in the vicinity of the physical structures being captured. Furthermore, since the resolution is only enhanced locally at the features, with coarser grids near successively uniform flow regions, the computations with such grids consume significantly less computer resources than global refinement. There are substantial savings in both CPU time and memory. The technique is also devoid of node-entanglement phenomenon, since the nodes are not allowed to move and the topology of the base grid is preserved in the finest meshes. Alternately, the skewness of the finest grid can be no worse than that of the initial coarsest mesh. The disadvantage of the approach is that the logic of an adaptation procedure is generally complicated and the resulting unstructured data-base is prone to errors. Such an approach demands expertise on the part of humans and sophistication on the part of the computers.

The spatial adaptation technique employed here belongs to the local embedding class. A multiple-grid technique is not used, since it is inappropriate for unsteady problems, and thus meshes at various coarseness levels are part of the same global grid. Since the initial and subsequent grids at any moment can be unstructured, a block grid technique is useful to generate initial grids for complicated geometries. The block grid approach allows the patching together of simple algebraic grids that conform to local boundaries in various regions, but unlike the zonal approach care must be taken to match the nodes on common interfaces.

Before proceeding with a description of division, collapse and other grid manipulations, the managing data structure or pointer system that controls spatial grid alterations will be briefly introduced. Familiarization with the data-structure facilitates the understanding of spatial grid manipulations. Only the data-structure pertaining to two spatial dimensions and for spatial adaptation will be discussed here. A more complete detail for spatial data structure, temporal adaptation, and chemistry pointers appears in Appendix C.

5.2 Spatial Data Structure

The familiar (i, j) indexing system used for structured grids cannot be used with local embedding spatial adaptation, since such a procedure generally destroys the "structure" of an existing mesh. An unstructured pointer system lends itself to effective refinement strategies. However, it suffers from inherent limitations, such as the need to store connectivity arrays and the use of gather-scatter operations on vector machines. Furthermore, the use of a number of algorithms, such as approximate factorization (Beam and Warming) and splitting methods (ADI), which were originally developed for structured grids can not be implemented on unstructured grids [87]. The importance of an efficient spatial pointer system for rapidly changing unstructured mesh in unsteady flow cannot be overstated; the pointer system described in this thesis is geared towards such efficiency. Once the grid structure is defined through a pointer system, a general solver can be implemented in terms of these pointers and the integration can proceed on a cell by cell basis and in any arbitrary order. This separation of grid structure from the flow solver allows creation of an efficient and modular approach.

The assignment of pointer systems to define the connectivity of objects in an unstructured grid is not unique; it depends upon the type of grids (triangular, quadrilateral, *etc.*), and the amount of detail desired (more flexibility implies more data storage) [33,57,79,99,133]. The spatial pointer system used here is very similar to those of Usab [133] and Dannenhoffer [33].

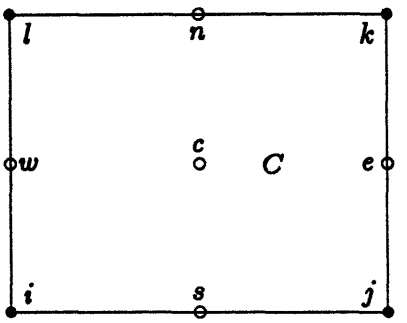

Figure 5.1: Node pointers for a given cell C.

The cell-to-node connectivity array defines the linkage of a given cell to its nodes and parent cell. For example, the nodes $i, s, j, \ldots, w$, in Figure (5.1), are pointed to by the cell-to-node array once the cell number C is known. The array in the present algorithm has ten pointers for each cell in the domain. The filled circles denote corner nodes which are always present while empty circles correspond to nodes which may or may not exist. None of the center and middle edge nodes exist for a given cell if it is undivided and does not border a divided cell. For unsteady flows, without a multiple grid technique, the cell numbers and corresponding arrays for divided cells are no longer needed and these assignments can be reallocated to the new cells that are created by the division process. However, to maintain generality, the divided cell numbers are retained even for unsteady calculations. This makes the book-keeping somewhat easier, since division of each cell in a two-dimensional domain increases the total number of cells by four, and the opposite holds for the fusion of cells. Hence the net difference of total number of cell before and after the spatial adaptation cycle is a multiple of four. The retention of divided cells means that for unsteady flows a *linked list* consisting of only undivided cells need be maintained for an efficient integration procedure.

The cell-to-node array has its usefulness when integration proceeds on a cell by cell basis and each cell increments to the changes that are accumulated at its respective

nodes. In order to avoid an expensive search procedure a reverse array pointer, namely, a node-to-cell array, is needed to specify the cells surrounding a given node. This array has four pointers for each node and is constructed such that if all four pointers of a given node are non-zero and unique, it is a common interior node; however, if the four pointers are non-zero and non-unique then the node is an interior middle edge node of a spatial interface.

In addition to the arrays that imply connectivity of nodes and cells, simple node-arrays and cell-arrays are needed for other manipulations, since nodes and cells may be numbered arbitrarily. The node-arrays contain geometry information, state vectors and some other variables at all of the computational nodes. The cell-arrays hold information pertaining to some or all of the cells in the computational domain. This may contain, for example, the refinement parameter values for undivided cells, the spatial level pointers of each cell *etc.*

Link-lists are needed to hold information pertaining to specific cells or nodes and these may be assigned in any arbitrary order. For example link-lists are needed to hold those cell numbers which must be divided (or fused) in the subsequent adaptive cycle. Boundary-Arrays contain information pertaining to the nodes on the domain boundaries. This is needed to apply boundary conditions, perform interpolation functions and facilitate grid adaptation near the boundaries.

5.3 Detection of Flow Features

One approach to detect flow features examines the first differences of a single pre-selected criteria variable [34,99,106,119]. A typical choice for this involves density differences since density appears as a factor in each element of the state vector; furthermore, density differences are present for most flow fields including shocks, contact discontinuities, *etc.* It is clear that for a system of N_e equations it would be expensive to examine every state vector component $U(k), k = 1, \cdots, N_e$ to define the necessary spatial and temporal resolution. However, use of a single variable might be insufficient when

different regions are characterized by different physical gradients. For example a concentration shock or contact surface may occur in one location with a small density gradient concurrently with a classical shock elsewhere without mass fraction gradients. If only one criteria variable is used some features may not be resolved adequately. Therefore a multi-variable approach is suggested with a special form for the differences.

5.3.1 Type of Differences

The types of differences used to detect features in spatial adaptation procedures are not unique. Kallinderis [71] has used divided and undivided first differences in viscous and inviscid regions of flow. Dannenhoffer [35] has used undivided first and second differences for this purpose. The undivided first differences can be interpreted as first order derivatives in the computational domain for unit cell dimensions, a similar statement can be made about the second order derivatives. Even when the type of difference is decided its numerical form may differ depending upon whether the differences are node based or evaluated on cells. Consider, for example, the first difference of density, $\Delta\rho$, in one spatial dimension. From Figure (3.1), the value at node j is

$$2\Delta\rho_j \;=\; \rho_k - \rho_i$$

provided that the dimensions of cells surrounding the node j are comparable; however, if cell C is twice as long as cell B, *i.e.*, node j represents a spatial interface, then the appropriate difference at this node is

$$2\Delta\rho_j \;=\; \frac{1}{2}(\rho_j + \rho_k) - \rho_i.$$

Thus the difference at a node can be complicated by the introduction of spatial interfaces. This situation becomes further complicated in two spatial dimensions where different kinds of spatial interfaces can exist. Furthermore node based differences have to be appropriately modified near physical boundaries. This also has the disadvantage that once a node is flagged as having a value of *refinement parameter* more than some *threshold limit* the cells surrounding this node must be scanned for possible division. It is generally unclear which cell has contributed most to the difference for a given node.

The density difference for cell C in the same figure is

$$\Delta\rho_C = \rho_k - \rho_j$$

which is clearly irrespective of any spatial interface location and does not have to be modified near physical boundaries. Since it is the cells that are divided or collapsed, it is natural to evaluate differences based on cells. These differences not only avoid complications due to grids but also can be evaluated at a lower computational cost and are consistent with the philosophy of cell by cell integration for the adaptive procedures. For these reasons the present algorithm utilizes undivided first differences on only cells without centers which are stored in a link-list to be used for this purpose.

In two spatial dimensions the cell differences can be evaluated as changes along each of the computational coordinates. For a scalar variable ϕ these differences are ϕ_ξ and ϕ_η; and these particular forms may be useful if directional adaptation is desired. However, if the directionality is unimportant or is undesired then differences based upon specific directions must be modified to yield some other unbiased measure of property variation. An example of such a non-discriminating overall difference is

$$\Delta\phi = \sqrt{\phi_\xi^2 + \phi_\eta^2}.$$

Another example is

$$\Delta\phi = |\phi_\xi| + |\phi_\eta|.$$

For accurate computation of ϕ_ξ and ϕ_η middle edge nodes must be used whenever such nodes exist; otherwise appropriate interpolated values have to be used. This is computationally expensive since it involve IF-THEN clauses to find out if these nodes exist. The exact detail and form of the first differences is generally unimportant; they are seldom used in their original form and are often normalized to yield ***standardized values***. Furthermore, for most unsteady flows since it is necessary to adapt frequently an efficient differencing scheme must be selected. For these reasons such differences are not computed in the present code. The computational time can be minimized if only corner vertices of a cell are considered when evaluating differences. This significantly reduces computing time since corner nodes always exist and IF-THEN structures are

not needed. Consider cell C in Fig. (3.5); the cell value for some scalar variable ϕ is

$$\phi_C = \frac{1}{4}(\phi_i + \phi_j + \phi_k + \phi_l). \tag{5.1}$$

Four cell differences are

$$\Delta\phi_m = \phi_C - \phi_m \qquad \text{for} \qquad m = i, j, k, l \tag{5.2}$$

and maximum and minimum difference values for the cell are

$$\begin{aligned} \Delta\phi_{max} &= \max\{\Delta\phi_i, \Delta\phi_j, \Delta\phi_k, \Delta\phi_l\} \\ \Delta\phi_{min} &= \min\{\Delta\phi_i, \Delta\phi_j, \Delta\phi_k, \Delta\phi_l\}. \end{aligned} \tag{5.3}$$

Note that these values are positive and negative respectively for locally non-uniform flow regions. The cell difference $\Delta\rho_C$ is then set according to

$$\Delta\rho_C = \begin{cases} \Delta\rho_{max} & |\Delta\rho_{max}| \geq |\Delta\rho_{min}| \\ \Delta\rho_{min} & \text{otherwise.} \end{cases} \tag{5.4}$$

For a large number of cells one is justified to assume that the average of all such changes is approximately zero, since there is equal likelihood for a general cell C to acquire positive or negative values. However, no such assumption is made here. Nevertheless, it has been observed that the average value of such differences has always been six or seven orders of magnitude smaller than the corresponding standard deviation in all cases that have been examined. Note that if the maximum absolute value of these changes is assigned as the cell change value then the number of computations can be slightly reduced; however, the average of the differences will be non-negative and will definitely have to be computed for the approach described below.

5.3.2 Multi-Variable Approach

Let Q^c denote the spatial criteria variable vector for a general cell c; the components $(q_1, q_2, \cdots)^c$ of this vector form the first differences of selected variables as indicated in the preceeding subsection. Thus if density is used as one criterion then $q_1^c = \Delta\rho_c$ and if mass fraction Y of some species is used as a second criterion then $q_2^c = \Delta Y_c$ and so on.

The mean value vector of Q^c over all the cell values is denoted by $(\mu_1, \mu_2, \cdots)$ which may be approximately zero. Once all the elements of the vector Q are determined for each cell, the variance-covariance matrix $\Sigma = \{s_{ab}\}$ is computed, where

$$s_{ab} = \sum_{c=1}^{N_c} \frac{1}{N_c}(q_a^c - \mu_a)(q_b^c - \mu_b) \tag{5.5}$$

and N_c is the total number of undivided cells in the domain, and the indices a, b vary between 1 and N_q which denotes the total number of components in the spatial adaptation criteria vector. The sample variance s_{aa} provides a measure of spread of data for observations of component a, whereas the sample covariance s_{ab}, for $a \neq b$, provides a measure of linear association between the observations of the components a and b. The correlation coefficient between these variables is

$$C_{ab} = \frac{s_{ab}}{\sqrt{s_{aa} s_{bb}}}. \tag{5.6}$$

If large and small observations of one variable occur respectively in conjunction with large and small values of a second variable then the sample covariance will be positive and the correlation between the two variables can be measured by the closeness of the correlation coefficient to +1. If large values of one variable occur simultaneously with small values of another variable and *vice-versa*, their sample correlation will be negative and the two variables will be inversely correlated. If there is no particular association between the values of the two variables, the correlation coefficient will be nearly zero. To accelerate the adaptive process one can assume that $s_{ab} = 0$ when $a \neq b$ for suitably chosen variables; however no such assumption is made for the illustrative examples shown here. Next a single scalar criteria variable is computed, which lumps the effects of the multi-variable components of Q for each cell and has the form

$$r^2 = r^2\Big|^c = (Q - M)^T \Sigma^{-1} (Q - M) \tag{5.7}$$

where Σ^{-1} is the inverse of the variance-covariance matrix and M is a diagonal matrix with entries equal to the mean values $(\mu_1, \mu_2, \cdots)$. The superscript c is omitted here for simplicity. This scalar variable will be referred to as the *refinement parameter.* The above reduces to the familiar form $r = (q - \mu)/\sqrt{s_{11}}$ for a single variable situation. The inverse of the variance-covariance matrix will not exist for spatially uniform flow

fields and in these cases there is no need to perform adaptation. However, appropriate measures must be taken in the software itself to avoid adaptation in such cases; in the STAR code if this matrix is determined to be ill-conditioned the refinement parameters of all the cells are simply set equal to zero. The contours of constant r^2 values for distributions in the space of N_q dimensions are hyperellipsoids defined by the Q values. In particular for a two dimensional space, the equation of an ellipse in (q_1, q_2) coordinates is

$$(q_1 - \mu_1)^2 s_{22} - 2(q_1 - \mu_1)(q_2 - \mu_2)s_{12} + (q_2 - \mu_2)^2 s_{11} \;=\; r^2(s_{11}s_{22} - s_{12}^2). \qquad (5.8)$$

On the standardized scales of $q'_a = (q_a - \mu_a)/\sqrt{s_{aa}}$ this becomes

$$q_1'^2 - 2C_{12}q'_1q'_2 + q_2'^2 \;=\; r^2(1 - C_{12}^2) \qquad (5.9)$$

which represents the equations of an ellipse. This reduces to a circle if the variables are uncorrelated.

Equation (5.7) provides a meaningful distance norm for data Q^c from its mean value in the case when the variabilities in different components are different and when some or all of these components are correlated. This measure removes the effect of inter-correlations between individual components instead of merely summing up the individual contributions. The standardized variables allow for an unbiased spread of data. This has the advantage that spatial domains characterized by different kinds of scales can be adapted by using a single refinement parameter that takes into account the variability of all components and multiple components of refinement parameters are then eliminated. Thus the same approach may be used to adapt, for example, in viscous and inviscid regions.

5.3.3 Threshold Values

A *divide threshold limit* R_d is a value of the refinement parameter such that any cell with $r^2 > R_d$ will be considered for possible division. Two kinds of divide threshold limits, R_{d1} and R_{d2}, are considered here; the first is assumed *apriori* whereas the second is computed based upon the current distribution of refinement parameter values for each

cell. The limit R_{d1} is user supplied and allows evasion of the cell division procedure when the flow field is globally uniform or when the gradients are reasonably mild. The second threshold limit is selected from histogram records as the value corresponding to a specific fraction C_{f_d} (usually 20%) of cells for which the refinement parameter is more than this limit. An inverse procedure (*i.e.*, finding R_{d2} from C_{f_d}) is needed for the determination of this value. For this purpose the minimum and maximum refinement parameter values are first determined over all the cells

$$\begin{aligned} R_{max} &= \max\left\{ r^2\Big|^c \,, c = 1, 2, \cdots, N_c \right\} \\ R_{min} &= \min\left\{ r^2\Big|^c \,, c = 1, 2, \cdots, N_c \right\}. \end{aligned} \tag{5.10}$$

Next the refinement parameter values are segmented into intervals of constant length and the number of cells (frequency) within each interval is counted. Thus if the total number of segments is n, the interval size of the segment is $\Delta R = (R_{max} - R_{min}/n)$ and the i^{th} segment or bin is given by

$$[R]_i = [R_{min} + (i-1)\Delta R, R_{min} + i\Delta R] \qquad \text{for} \qquad i \in [1, n]. \tag{5.11}$$

The fraction f_i of cells with the refinement parameter values in the i^{th} bin is found from the number of cells with r^2 values in this segment. The distribution of frequency f_i versus the refinement parameter is generally similar to a normal distribution curve for a large number of cells. The cumulative frequency C_{f_i} is determined to be the overall fraction of all cells with a refinement parameter value exceeding that of the i^{th} bin and is given by

$$C_{f_i} = \sum_{j=1}^{n-i+1} f_{n-j+1} = \sum_{j=i}^{n} f_j. \tag{5.12}$$

Now that a one-to-one correspondence between the cumulative frequency C_{f_i} and the mean value R_i of the i^{th} bin is established, the value R_{d2} can be obtained as the value corresponding to a pre-defined fraction C_{f_d} through linear interpolation between the appropriate bins. A single threshold value,

$$R_d = \max(R_{d1}, R_{d2}) \tag{5.13}$$

is then used as the decision basis for cell division.

In the case when $R_{d2} > R_{d1}$, it is unnecessary that the total fraction of the divided cells will be exactly C_{f_d}, since some cells which had been marked for resolution may not actually be divided. The cells are not divided if the *spatial level* of the subcells pertaining to a marked cell would exceed some user supplied maximum level. Furthermore the cells are not divided if the difference between any two contiguous cell levels would exceed unity.

The decision basis for cell merger, R_c, is set to be between 20 to 40 percent value of the divide threshold value. When the associated refinement parameter diminishes on a previously refined grid, and becomes less than the merger critical limit, those contiguous grids may be collapsed while making certain that the cells to be merged are from the same parent cell. Cells also are not merged if the difference of levels between the parent cells and its neighbors would exceed unity. The initial (coarse) global grid is kept stagnant by insisting that the coarsest cells (spatial level zero) be never merged to a coarser state, no matter how smooth the evolving solution proves to be.

5.4 Grid Division

Once refinement parameter values r^2 are computed for all individual cells and threshold values R_d and R_c are determined, all cells with $r^2 > R_d$ are flagged for possible division whereas those for which $r^2 < R_c$ are flagged for possible fusion. Before the actual cell division procedure is invoked for the cells to be divided, the link-list containing the cells to be divided is extended to include cells in the regions neighboring the one which is marked for further resolution. The logic for the determination of extended cell regions is deferred until a later section.

Before a particular cell can be divided a number of other conflict rules governing subdivision are examined. The simplest rule examines the remaining space in the data base for availability so as to place additional pointers which the newly created cells would demand. If the data base is not saturated further evaluations are allowed. This rule does allow redistribution of grid points once the data base is saturated. Next the

spatial level of the cell to be divided is examined and verified to be less than a user-supplied value. Without this rule the cells would be divided indefinitely near regions which propagate slowly. Note that for steady state situations, this rule may not be needed since the grid may be adapted only a few number of times.

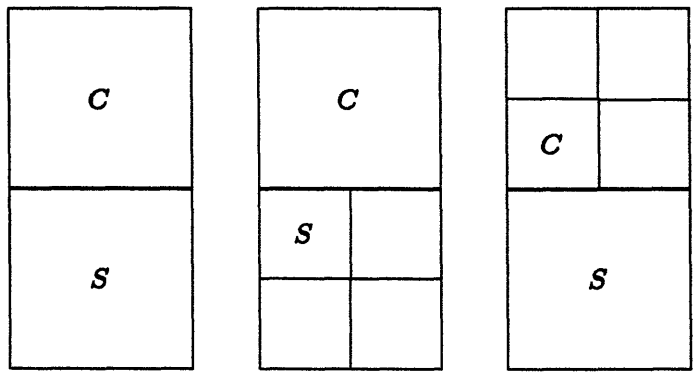

Figure 5.2: Three possible situations for spatial level differences.

Another rule examines the difference between spatial levels for the cell to be divided and any of the neighboring cells, and aborts the division process if this difference is such that further division will cause the cell volumes to differ by factors of more than four. Consider the three possible permutations, shown in Figure (5.2); the level L_C of the cell C to be divided and the level L_S of a southern neighbor S are examined. The division is allowed to occur only in cases (a) and (b). This rule is designed to avoid the substantial stiffness that the spatial grid would otherwise introduce due to the disparity in the cell volumes. Such stiffness will subsequently be referred to as *spatial level stiffness.*

After all preliminary tests are completed, a node is created at the centroid of the cell, and dependent variables are set equal to the average values of the corner nodes. If the nodes at the face midpoints do not already exist, they are created, and new nodal values for the node-arrays are interpolated from nearby face nodes. Similarly four new fine cells are created with cell numbers exceeding the previous value of the total number

of cells. All pertinent arrays are adjusted appropriately to account for additional nodes and cells. The reader is referred to Appendix C for additional details.

New cells are tagged to indicate that these cannot be collapsed for three more spatial adaptation cycles. This rule is designed to create a lag between the most recently divided and subsequently fused cells. It is possible that a cell to be divided lies within a *buffer zone* and is awaiting the arrival of a feature, but the feature might be delayed due to stringent time-step restraints elsewhere in the domain and might not reach the divided cell until after two or three time-strides. Thus, in this situation, if the cells are allowed to fuse in the second cycle, they may have to be redivided in the third cycle, and this rule simply defers this kind of situation.

The boundary pointers also are examined to see if special interpolation functions are needed to define the geometry at the middle edge node that conforms to a special solid boundary surface. For example, a quadratic form may be used for a circular arc bump and a cubic spline for other surfaces.

5.5 Grid Collapse

The reverse procedure that removes subcells is slightly more complicated than the cell division process. For a given cell number contained in the link-list of the cells to be merged there must appear exactly three other cells with the same non-zero supercells that have been flagged for fusion; otherwise the fusion process will not commence. Once located, the four subcells are arranged according to the relative cell number order in which they were created, so that reverse manipulations can be started.

In order to avoid spatial level stiffness, the level pointers of cells that neighbor supercell of the subcells to be fused are examined. If the difference of levels between these would exceed unity due to the application of the fusion process, the process is aborted.

There are situations for which it is known *apriori* that spatial resolution may be

permanently needed in certain locations. For example, in the vicinity of external or internal fuel-injection, one may want to maintain fine grid resolution even when the prevailing gradients of the resolution parameter become small momentarily for a certain span of time. This can be accomplished by tagging the cells in such regions to be "permanent residents" and therefore not allowed to collapse. If such a tagged cell is detected during the collapse procedure, the process is aborted.

After all preliminary tests are completed, the center node of the supercell is flagged for removal. Those side nodes which are not needed by the neighboring cells are also flagged for removal. To avoid gaps which would otherwise be created by removing the fine cells, such cells are replaced by the last four cell numbers in the domain. The situation becomes complicated if one or more of the last four cells is to be locally divided. Hence care must be exercised in performing the realignment of all the pointers between these two sets of cells. The reader is referred to Appendix C for details of this procedure from a coding perspective.

5.6 Extension of Spatially Resolved Region

For some unsteady flow situations, it is necessary to extend the spatially resolved region by a certain number of cells in the direction of propagation of flow features. This ensures that features remain within the spatially resolved region during a subsequent time-stride unit. For example, if a moving shock is being tracked and temporal adaptation is being used to allow advancement of cells with varying time-steps, it is possible to foresee that the shock may emerge from the edge of the resolved region by the time all cells in the time-stride sequence are integrated. It would be efficient to take into account the direction of motion of a feature when allocating a buffer zone of resolution, but such techniques would involve very complicated logic. For that reason the present code simply includes buffer zones applied in all directions to the existing spatially resolved regions.

Although the cells to be divided may exist at various spatial locations in the domain

and may be part of a number of distinct clusters, the set of these cells are referred to as the ***detected cluster*** to distinguish them from cells in the ***buffer zone.*** The total number of cells that extend across a detected cluster on each side or the width of the buffer zone is denoted by N_x. For the purpose of extension the cells in the detected cluster are examined to locate ***boundary*** cells of the cluster, and their edges or corners are ***painted*** appropriately to indicate extension through them. The buffer zone is added in distinct layers, and the total number of these is N_x.

If a cell in the detected cluster has a neighbor at a higher spatial level (or alternately is divided) then it is unnecessary to extend ***through*** a corresponding ***edge*** or ***corner.*** For example, if a southern node exists (so that there are two southern cells), then the extension through the southern edge is not needed. Similarly, if the north-west neighbor cell is at a higher level, then the extension through the north-western corner is not needed. After this examination, the neighboring cells which are possible candidates for the buffer zone are checked in the detected cluster. The cells which are located in this cluster cannot form the buffer zone and the corresponding edge or corner of the cell under consideration is painted for no extension. At this point a list of eligible candidates for the buffer zone can be formed, and attention can be focused on the next cell in the detected cluster. Subsequent candidates for the buffer zone would have to be checked in both the detected list and the current list of candidates.

The candidate cells collected so far form an outer boundary to the detected cluster or the first layer of the buffer zone. Subsequently only the cell in the first layer must be examined for further extension if N_x exceeds unity. Furthermore only those edges or corners of these cells should be examined which had not been painted in the previous pass.

If N_x is greater than unity, the cells in the first buffer layer are examined for possible extension and the whole process is repeated to form the next layer of the buffer zone. This procedure is continued until the desired number of layers is formed.

Once all extensions are completed, the cells marked for possible fusion are examined and any cell that appears in the overall buffer zone is removed from the fusion list.

This provides a more biased and conservative approach towards the fusion of cells. The reader is referred to Appendix C for coding details.

5.7 Islands and Voids

An *island* is defined as a single divided cell which is bordered by undivided cells at the same spatial levels. A *void* cell is one which has any of the following properties

- at least three divided edges
- at least two divided edges and is on a physical boundary
- two divided edges and is contiguous to a similar cell.

It is generally helpful to remove the abrupt changes that are caused by islands and voids. Although such a procedure is not essential for the spatio-temporal algorithm, such grid features are detected and removed for aesthetic purposes. Their occurrance in the overall grid is simply distracting. Examples of these features are shown in Figure (5.3). Note that an overall row of cells embedded in an otherwise coarse region is tolerated, but a row of void cells is removed by carrying out multiple passes of the void detection procedure as described in detail in the subroutine `A2VOID`.

5.8 Block Grid Generator

The generation of initial grids for complex flow geometries can be a difficult task. The grid generation even for simple flow fields with multiple embedded solid objects can be troublesome. For an initial grid generation an interactive multiple-block generator has been developed as part of an effort involving the current research.

A block grid method subdivides the flow field domain into regions known simply as blocks. The topology of one block has no bearing on the rest of the blocks, excepting

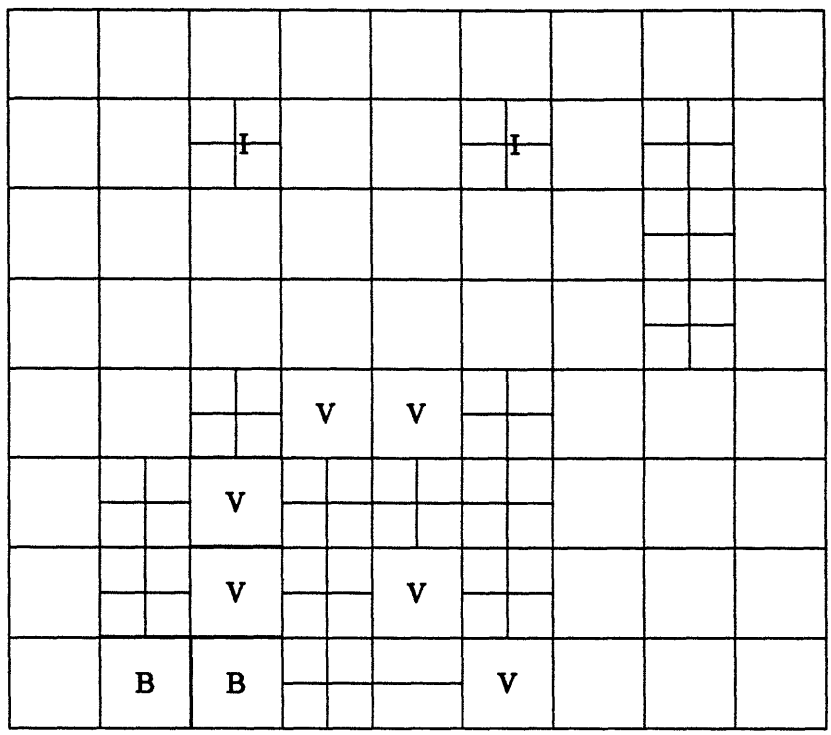

Figure 5.3: Portion of a grid with islands marked by I and voids marked by V; cells marked by B are those which become void cells in a second pass.

that there must be a node-to-node matching across the interfaces of contiguous blocks. The block grid approach is similar to a zonal approach, but since the nodes of the contiguous blocks coincide at the block interface, the need to perform complicated flux balances at the interfaces is eliminated. The advantage of the approach is that a clever choice of block boundaries can reduce complex flow field regimes into smaller numbers of less complicated regions and hence the overall grid generation becomes a simpler task. The block grid approach ties in neatly with the finite volume implementation. Since the integration is carried out on the basis of flux balances through the differential volumes, the size, shape and skewness of the grids is of less consequence compared to the usual finite difference approaches.

Literature on the subject-matter has not revealed any reliable automatic procedures for subdividing an arbitrary domain into simpler blocks. It is complicated in the sense that it involves inherent knowledge of the physical domain, and that subdivision into simpler computational domains is not unique. The logic is further complicated by additional zoning constraints for specific applications. However, once the total number of blocks is decided and their physical locations into a final assembled grid is determined, it becomes a simple matter to fill in the internal mesh for each block and align nodes on the contiguous block surfaces.

For the block grid generator developed here, the total number of blocks, the geometry of each face of the block in terms of cubic polynomials and the number of boundary points on two adjacent faces must be specified. The interior mesh for each interior point is then generated by an algebraic grid and the connectivity arrays for each cell in each block are determined. This means that additional nodes will exist at the time of assembly of the overall grid when the points on the contiguous boundaries coincide. These multiply defined nodes are marked for deletion and the connectivity arrays of the boundary nodes are examined and adjusted for consistency in the data structure. The user is able to view this assembly interactively at various stages and could request the program for certain changes. For example, the user may move nodes in certain regions, subdivide meshes or fuse four adjacent cells, *etc.* A listing of the interactive grid generator, `GNBLOC`, is provided in Appendix D.

Chapter 6

Temporal Adaptation

The concept of utilizing variable time-steps for solving time-accurate transient problems is developed here. The chapter begins by examining the factors which limit the computational costs and the ways in which these costs can be reduced. The classical integration scheme for which a global minimum time-step applies will be referred to as *one-step explicit* or simply Ni scheme. The scheme permitting variable time-steps will be referred to as *multi-step explicit* or simply adaptive scheme. The issue of temporal resolution is discussed in Section (6.2) for frozen and reacting situations in both one and two spatial dimensions. The concept of temporal adaptation is developed for one dimensional systems in Section (6.3) followed by an illustrative example in Section (6.4). The temporal adaptation concept is generalized to include larger time-stride units in the last section.

6.1 Motivation

In chemically reacting flows, the computations of chemical kinetic terms is often more expensive than evaluations of convective and/or diffusive transport terms. The cost increases with the number of species, the number of reactions connecting these species, the number of spatial cells and the inverse of the time-step size. For flame and detonation simulations the overall calculation may take two or more orders of magnitude longer compared to frozen flow situations. Calculations may also be costly due to stiffness introduced into the equations by the finite rate chemical kinetics which is necessary to describe the physical situation. These factors form a basis for a need to generate more efficient and accurate algorithms for solving reacting flows.

The calculations involving diffusive transport terms are generally not expensive, for a single cell, compared to the overall manipulations of source terms for typical chemical reaction systems. However, the transport phenomenon demands additional resolution near boundaries, interface of two streams, *etc.*, and the overall computational costs increase drastically with the added number of cells. The computational overhead can be somewhat reduced by considering Euler equations on relatively coarser grids and neglecting these fine features whenever it is reasonable to do so. The computational costs can be reduced further by avoiding the expensive evaluations of source terms and their Jacobians in the regions of embedded frozen flows in an otherwise reacting simulation, since in these regions the source terms are negligible compared to the corresponding terms in the relaxing regions of the domain. Generally chemical reactions proceed at a negligible pace if the temperature is below a "threshold" value. For example, the combustion of hydrogen in air is negligible below about 1000 Kelvin. Hence, whenever static temperature is below the threshold limit the change due to chemical species equations need not be evaluated and corresponding state values may have to be updated so as to reflect only a change in global density while leaving the mass fraction values unchanged.

When the reactive equations are stiff in the sense that numerical stability rather than accuracy dictates the time-steps, then an implicit scheme can be used to partially alleviate the computational overheads. However, for unsteady flows, if there are local rapid chemical adjustments, the time-steps must be appropriately small to resolve the features. These are generally changing patterns of resolution requirements as the rapid transients form, gather strength, interact and deform other flow features and eventually decay in different periods and positions. Hence there are conflicting requirements on unsteady reacting flows in the sense that for efficient advancing time-steps may have to be reduced in certain portions of the space-time domain where adjustments occur and a utilization of longer time-steps be made where there are negligible temporal gradients.

Just as different spatial resolutions are allocated at different locations of a spatial grid to achieve CPU time gains, it would be beneficial to take advantage of the large spatial variations of time-steps for reacting flows. In fact gains due to utilization of

different time-steps can even be achieved for unsteady frozen flows if there exist substantial variations in spatial cell volumes, which indeed may well be a result of spatial adaptation. An efficient time-differencing technique is developed in this chapter that makes possible advancement of cells on a step-size which is a multiple of a global minimum time-step. Without this technique the severe and costly constraint associated with a globally minimum time-step would be applicable and computational costs would be literally immense. In this technique the cells with the same time-step are integrated and updated together on different integration passes of the temporal adaptation cycle but the majority of small time-step cells fall in only a small portion of the overall space/time domain. Once all integration passes are completed for each time-stride unit, all nodes in the domain arrive at the same time-station.

6.2 Temporal Resolution

6.2.1 One Spatial Dimension

For unsteady flows temporal changes must be monitored so as to maintain sufficiently small time-steps for adequate local resolution and stability. To develop a criterion for temporal resolution first consider the governing Equations (2.45) in one spatial dimension and for simplicity restrict attention to a single species equation. If the magnitude of the source term W is relatively small, or alternatively if the chemical time-scale τ is large compared to the convective time-scale, then the temporal resolution Δt, at which the equations are advanced, is dictated by the CFL restriction for explicit schemes, *viz.*

$$\Delta t_{cfl} \leq \frac{\Gamma \Delta x}{|u| + a_f} \tag{6.1}$$

where $\Gamma \leq 1$ is the CFL number and a_f is the local frozen speed of sound. This constraint indicates coupling of the time-steps with the spatial resolution. For such problems $\partial U/\partial t$ is essentially the order of $\partial F/\partial x$ and W is small compared to the other terms, *i.e.*,

$$\frac{\partial U}{\partial t} \sim \frac{\partial F}{\partial x}, \qquad W \ll \frac{\partial F}{\partial x}. \tag{6.2}$$

Case	Terms
1	$\frac{\partial F}{\partial x} \sim W$ and $\frac{\partial U}{\partial t} \ll \frac{\partial F}{\partial x}$
2	$\frac{\partial U}{\partial t} \sim W$ and $\frac{\partial F}{\partial x} \ll \frac{\partial U}{\partial t}$
3	$\frac{\partial U}{\partial t} \sim \frac{\partial F}{\partial x} \sim W$

Table 6.1: Balance of terms for one-dimensional, one-component system when the source term is relatively large.

For large values of the source term there are three possibilities as indicated in Table (6.1). The first case is analogous to a steady state problem; large spatial gradients are present but the variation in time is negligible. The need to maintain adequate spatial resolution is obvious. However, there is no need to resolve the flow features within a time-scale less than that dictated by the CFL restriction, and hence a source implicit scheme is justified.

For the second case flow features must be resolved in time and a time-step smaller than that dictated by the CFL restraint may be required. In such cases the temporal gradient $\partial U/\partial t$ must be modelled carefully; the magnitude of $\partial U/\partial t\ \Delta t$ may need to be restricted so that only small changes occur for each time-step. This will yield a smooth variation of the state vector with time.

For the third case both spatial and temporal rates of change are comparable; and resolution is needed in both space and time. For this case the time-steps may not have to be as small as in the previous case since the large source term may be partially balanced by a spatial flux gradient. The third and second cases are similar so far as temporal resolution is concerned and as indicated in the subsequent the same criterion

can be applied. Consider the cell C in Figure (3.1) and limit the cell change according to the following criterion

$$\Delta U_C \approx \left.\frac{\partial U}{\partial t}\right|_C \Delta t_{res} \leq \Delta U_{max} \tag{6.3}$$

here the time-step indicates the resolution requirement and ΔU_{max} is the maximum allowable change for the species equation (in fact this could be applied to other equations also). A threshold criterion for this maximum allowable change will be discussed later. The change for cell C is given by Equation (3.11) and can be written as the product of a *driving force*, D, and cell time-step, *i.e.*,

$$\Delta U_C = D\Delta t_C = \left(W_C + \frac{F_j - F_k}{\Delta x_C}\right)\Delta t_C. \tag{6.4}$$

Note that the species density will increase if the driving force is positive and *vice versa.* Comparing the last two equations yields the restraint for time-step resolution

$$\Delta t_{res} \leq \left|\frac{\Delta U_{max}}{D}\right| = \left|\frac{\Delta U_{max}\Delta x_C}{W_C\Delta x_C + (F_j - F_k)}\right|. \tag{6.5}$$

This again indicates a coupling of spatial and temporal resolutions. The resolution requirement, Δt_{res}, may or may not exceed the stability requirement, Δt_{cfl}, and the actual time-step is

$$\Delta t = \min\{\Delta t_{res}, \Delta t_{cfl}\}. \tag{6.6}$$

Note that in the familiar limit of non-reacting uniform flow $\Delta t_{res} \longrightarrow \infty$ and the stability requirement is governing. On the other hand, large W_C for uniform flows implies $\Delta t_{res} \ll \Delta t_{cfl}$ and the expected problem of stiffness. For this case, if the uniform flow conditions persist, the flow will start approaching the equilibrium limit and larger time-steps could be taken subsequently since the overall change in the species density will diminish and the source term will itself become smaller. In addition to the drive towards equilibrium the flux gradients may emerge which may provide a balance with the source term and hence temporal gradients will diminish which will allow larger time-steps. As an example consider Figure (6.1) where a relaxation process start far away from equilibrium. Initially, the drive towards equilibrium is fast and it slows down at a later time. The relaxation may never approach identical equilibrium if substantial flux gradients exist. If the time resolution is held to a constant change in the species density

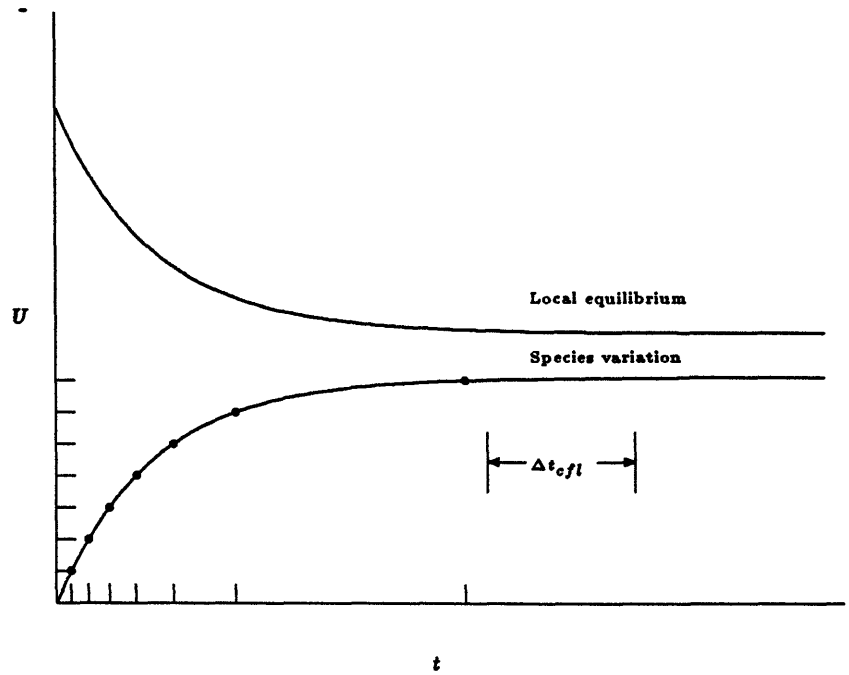

Figure 6.1: Allocation basis for resolution time-step.

as indicated by the circles on the relaxation curve (and the marks on the vertical axis), then it is clear from the ticks on the time axis that the time-steps would gradually increase as the slope tapers off to an increasingly smaller value. As the curve flattens out the time-step becomes infinitely large for this constant change model. It is possible that initially the CFL constraint might be less stringent compared to the resolution time-steps but it would eventually become more stringent as time increases. Note that during the initial transient Equation (6.5) may be as stringent as the stability restriction that would be dictated by an explicit scheme due to the chemical time-scales since the source term appears in the denominator. However, as time elapses the driving force decreases and larger time-steps can be taken. This is the essential modification to the implicit source approach [23] that allows unsteady computations. As asserted here, for time accurate descriptions a criterion such as Equation (6.5) is desirable for both explicit and implicit integration schemes, although for frozen flow this restriction may be of lesser consequence.

For the first case in Table (6.1), the balance between flux derivative and source term, *i.e.*,

$$F_j - F_k + \Delta x_C W_C \approx 0$$

implies that for finite, non-zero values of the threshold limit, the resolution time-step restriction approaches infinity and the CFL restriction governs, *i.e.*, $\Delta t_{res} \gg \Delta t_{cfl}$, as is typical for steady flow situations. For the second case the flux gradient is nearly zero and the resolution requirement simplifies to

$$\Delta t_{res} \leq \frac{\Delta U_{max}}{W_C}. \tag{6.7}$$

Thus very large values of the chemical source term imply very small values for the resolution time-step, which is then the most restrictive, *i.e.*, $\Delta t_{res} \ll \Delta t_{cfl}$. For the third case the two time-steps may be of the same order of magnitude. It is possible that all three types of balances may exist at different spatial locations and at different times in a given simulation.

A threshold criterion for the maximum allowable change in the species equation will now be suggested. For a non-vanishing species state variable, the maximum allowable change may be defined as a small fraction of the state variable itself. This is because the mass fraction variations for a non-inert species with high concentration may be proportionally larger (generally higher concentrations species react more). However, allowance must be made for near zero levels, for which infinitesimal time-steps are irrelevant. A suitable form for the threshold is then

$$|\Delta U_C| \leq \Delta U_{max} = \epsilon_1 U_C + \epsilon_0 \tag{6.8}$$

where the ϵ_i are small positive numbers. Effectively, the change is limited to a fraction of the state value excepting for vanishingly small levels. The threshold form utilized in Figure (6.1) corresponds to $\epsilon_1 = 0$.

Gear [54] has suggested restrictions on the growth of truncation error, in limiting the time-step for adequate resolution in the numerical solution to be below some set level for the integration of stiff ordinary differential equations. That approach becomes very complicated for partial differential equations and is not used here. Gear's approach for

temporal resolution is analogous to the approach taken by Berger [13] who uses truncation for reapportionment of spatial grids. Since for spatial adaptation Dannenhoffer [34,35] has shown that first differences of certain flow variables may be used instead of the truncation error, the same approach can be extended for temporal grids where the temporal gradients are kept small as indicated by Equation (6.3). Drummond *et. al.* [42] have used a simpler form in which $\epsilon_1 = 0$ in connection with Equation (6.7) and their representation corresponds to the second of the balancing situations in Table (6.1).

If the driving force D is positive and $U_C = \rho Y$ is small, the restriction imposed by Equation (6.8) may be unnecessarily severe. In such a case even when $\Delta U_C \sim U_C$ a reasonable resolution can result so long as the updated mass fraction is small compared to the maximum possible mass fraction Y_{max}. For that situation a reasonable maximum allowable change can be modelled as

$$\Delta U_{C_{max}} = \epsilon_1 \rho (Y_{max} - Y) + \epsilon_0 = \epsilon_1 (U_{max} - U_C) + \epsilon_0 \qquad \text{for} \qquad D > 0. \qquad (6.9)$$

When the driving force D is negative Equation (6.8) is appropriate. When the driving force vanishes, as in first case in Table (6.1), there is no need to restrict time resolution based on the species equation.

A pertinent question after the development of a temporal resolution basis for the one species equation relates to multiple-component reaction systems. Just as there is no need for spatial embedding to resolve every component of the state vector since they prove to be coupled, temporal resolution needs also may be based on only a few of the species that are present. Since the fluid mechanic time-scales are already resolved by the CFL restriction, a single dominant species that provides the resolution for a minimum time-scale associated with the chemistry may suffice. Another possibility would be to examine the current maximum change among all species and limit its change by restricting the time-step. However, that would involve the computation of driving forces for all species and would be computationally expensive. Hence the former approach was utilized here for its simplicity. Unlike the choice of spatial criterion variables for resolution (*e.g.*, density and any of the species mass fractions) that for temporal resolution is not obvious *apriori*. It should, however, correspond to a species which is expected to change most

rapidly and frequently. This species typically takes part in a large number of reactions and these reactions have large rate coefficients. The fact that exchange reactions are generally faster than dissociation reactions can be important in making this choice.

6.2.2 Two Spatial Dimensions

For two spatial dimensions the cell change ΔU_C for a cell C, as indicated in Figure (3.5), is given by Equation (3.46) in terms of the corner nodes. This can be used to define the driving force as

$$D = W_C + \frac{1}{A_C} \oint_C (F dy - G dx) \tag{6.10}$$

where the discretized flux balance in Equation (3.46) has been replaced by its continuous representation for simplicity. The resolution time step restriction, analogous to Equation (6.5), becomes

$$\Delta t_{res} \leq \left| \frac{\Delta U_{max}}{D} \right| = \frac{\Delta U_{max} A_C}{|A_C W_C + \oint_C (F dy - G dx)|} \tag{6.11}$$

for a pre-selected criterion variable. The maximum allowable change for the criterion variable is limited as (see Eqs. 6.8 and 6.9)

$$\Delta U_{max} = \begin{cases} \epsilon_1 U_C + \epsilon_0 & D < 0 \\ \epsilon_1 (U_{max} - U_C) + \epsilon_0 & D > 0 \\ \infty & D = 0. \end{cases} \tag{6.12}$$

The CFL restriction is given by Equation (4.2) and the current time-step allocated to a cell is the minimum of the resolution and CFL constraints.

6.3 Discussion of Temporal Adaptation

To motivate the development of variable time-steps for solving unsteady problems consider the following simple form of Euler equations in one spatial dimension

$$\frac{\partial U}{\partial t} = -\frac{\partial F}{\partial x}. \tag{6.13}$$

Once the concept is developed, it will be extended to include source terms for both one and two-dimensional situations. For the sake of demonstration, assume that the non-uniformity parameter ϵ, from Chapter 3, is identically zero for all nodes at which the scheme is applied and for which no artificial viscosity is needed.

Consider cells B and C surrounding the node j in Figure (3.1) for the explicit Ni scheme. As given by Equation (3.21), the overall change at node j is the sum of contributions from cells B and C, *i.e.*,

$$U_j^{n+1} = U_j^n + \delta U_{jB}^{n,n} + \delta U_{jC}^{n,n} + O(\Delta t^3) \tag{6.14}$$

where the superscripts on the change contributions indicate an evaluation on the basis of flux values at time-level (n). Specifically, the superscript (n,n) indicates that both nodes i and j of cell B use values at time-level (n). These change contributions for the above simplified model are

$$\begin{aligned} \delta U_{jB}^{n,n} &= \tfrac{1}{2}\left(I + \tfrac{\Delta t}{\Delta x_B} F_{U_B}^n\right) \Delta U_B^{n,n} \\ \delta U_{jC}^{n,n} &= \tfrac{1}{2}\left(I - \tfrac{\Delta t}{\Delta x_C} F_{U_C}^n\right) \Delta U_C^{n,n} \end{aligned} \tag{6.15}$$

in which

$$\begin{aligned} \Delta U_B^{n,n} &= \left(F_i^n - F_j^n\right) \tfrac{\Delta t}{\Delta x_B} \\ \Delta U_C^{n,n} &= \left(F_j^n - F_k^n\right) \tfrac{\Delta t}{\Delta x_C}. \end{aligned} \tag{6.16}$$

For the case involving both source terms and grid non-uniformities Equation (3.35) may be used for change contributions. This is shown graphically in Figure (6.2) where the change contributions from both cells B and C are based on the same time-level (n) as indicated by the upper circle at node j. The states for nodes i and k are not shown explicitly, but also are evaluated at the same time-level. The figure also shows the variation of a component of the state vector for node j as a function of time, although only discrete values indicated by the circles are available. The value predicted by the one step explicit scheme at node j after a time Δt, is indicated by the lower circle which is the state variable at time-level $(n+1)$. At another time-level $(n+c)$ for the change contributions in Equation (6.14), the order of accuracy remains the same; *i.e.*,

$$U_j^{n+1} = U_j^n + \delta U_{jB}^{n+c,n+c} + \delta U_{jC}^{n+c,n+c} + O(\Delta t^3) \tag{6.17}$$

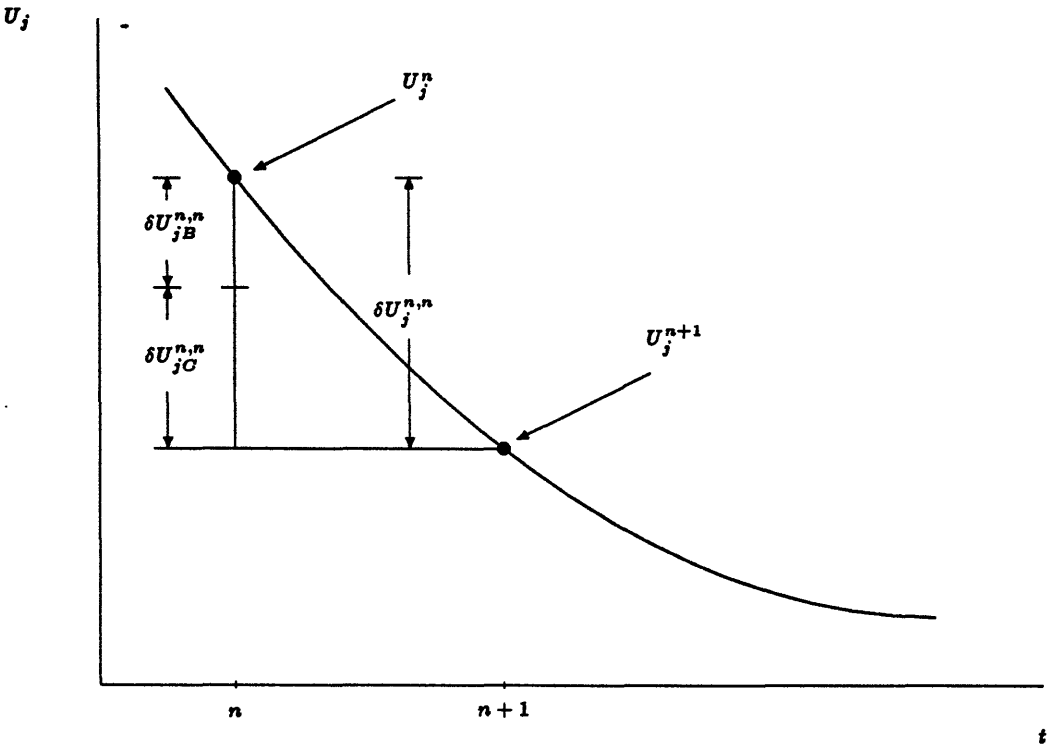

Figure 6.2: Graphical representation of the explicit Ni scheme.

where $(n+c)$ represents some time-level intermediate between (n) and $(n+1)$. In fact the change contributions from B and C can be evaluated at different time-levels, *i.e.*, one may consider different non-zero values c for these cells. If one integrates cell B based upon values (n,n) for nodes i and j and updates both nodes before actually integrating cell C, then an intermediate time-level $(*)$ is attained at node j, *i.e.*,

$$U_j^* = U_j^n + \delta U_{jB}^{n,n} \tag{6.18}$$

where the change due to cell B is given by Equation (6.15). Based upon state variables at that time-level $(*)$, the flux vector F^* is available, and the evaluation for the change contribution at node j due to cell C can be obtained out from

$$\delta U_{jC}^{*,n} = \frac{1}{2}\left(I - \frac{\Delta t}{\Delta x_C} F_{U_C}^*\right) \Delta U_C^{*,n} \tag{6.19}$$

where

$$\Delta U_C^{*,n} = \left(F_j^* - F_k^n\right) \frac{\Delta t}{\Delta x_C}. \tag{6.20}$$

This is shown graphically in Figure (6.3). The upper circle on the curve shows the

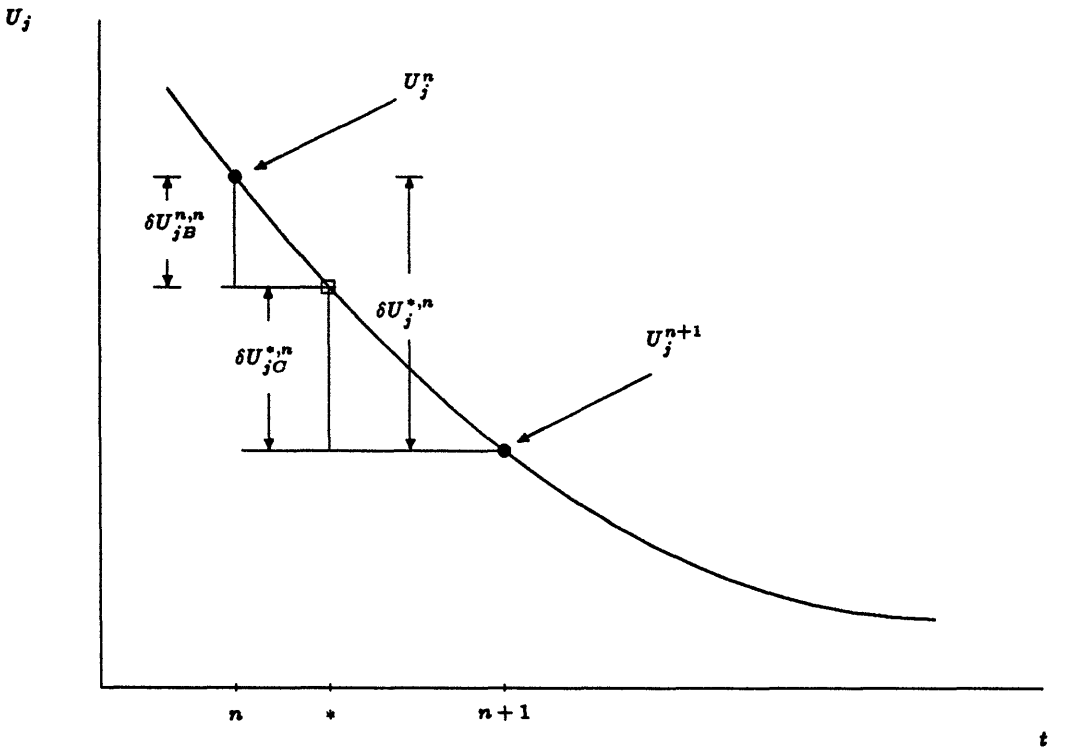

Figure 6.3: Graphical representation of single step predictor-corrector scheme.

change for cell B to be based on level (n), and the square shows that the change for cell C is based on level $(*)$. The overall change is due to their sum, *i.e.*,

$$U_j^{n+1} = U_j^n + \delta U_{jB}^{n,n} + \delta U_{jC}^{*,n}. \tag{6.21}$$

Note that the time level $(*)$ is not necessarily midway between (n) and $(n+1)$ and that its exact value for node j is of lesser concern for the current discussion, since primary interest is in the intermediate value of the state vector and not the time itself. Hence one can use Equations (3.22) for $\delta U_{jB}^{n,n}$ and $\delta U_{jC}^{*,n}$ if one only stores the values of state vector at various nodes and updates them as soon as the change contributions are computed.

The latter approach can be regarded as an explicit *predictor-corrector* scheme in contrast to the Ni scheme which is a single step explicit scheme. The two approaches do not yield identical results but differ only within the order of the scheme itself. Note that the latter approach would be computationally more expensive since updating has to be performed prior to the change determination for cell C; however, the updating process itself is very inexpensive since it involves only the addition operation and hence the overall increase in CPU time would be marginal. Let us now examine the conservation property of the predictor corrector scheme. The explicit scheme is conservative in the sense that the flux contribution from cell B to node j [*i.e.*, $F_j(B) = -F_j^n$] is the same as the flux contribution of cell C at node j [*i.e.*, $F_j(C) = +F_j^n$], hence

$$F_j(B) + F_j(C) = 0.$$

For the predictor corrector scheme, this is no longer the case since the flux contribution from cell B is $-F_j^n$ and that from cell C is $+F_j^*$. However, since the fluxes differ by second order in time, the conservation property has been compromised in favor of the beneficial temporal adaptation. Since the predictor corrector approach will be applied only at nodits, which form only a fraction of the nodes in the overall domain, the conservation property is still valid away from these nodes.

Suppose now that the cell C properties can be advanced at a time-step twice that of cell B as indicated in Figure (6.4). It is assumed that the time-step of cell B and those to its left is Δt_B, and for cell C and those to its right is $2\Delta t_B$. Hence node j in this figure is at a *temporal interface* or *nodit*. Nodes which are not nodits will be referred to as *common* nodes. In the previous two single step approaches node j was regarded as a common node. An integration and subsequent updating of all cells to the left of cell B would advance the time level to $(n+1)$ for all the nodes to the left of node j; and a similar process for cells to the right of cell C would advance the time level to $(n+2)$ for all nodes to the right of node j. The time level for the node j itself would be somewhere in between $(n+1)$ and $(n+2)$. Clearly to arrive at the same time level $(n+2)$ would require integrating cell B and those to its left twice as often compared to all the other cells.

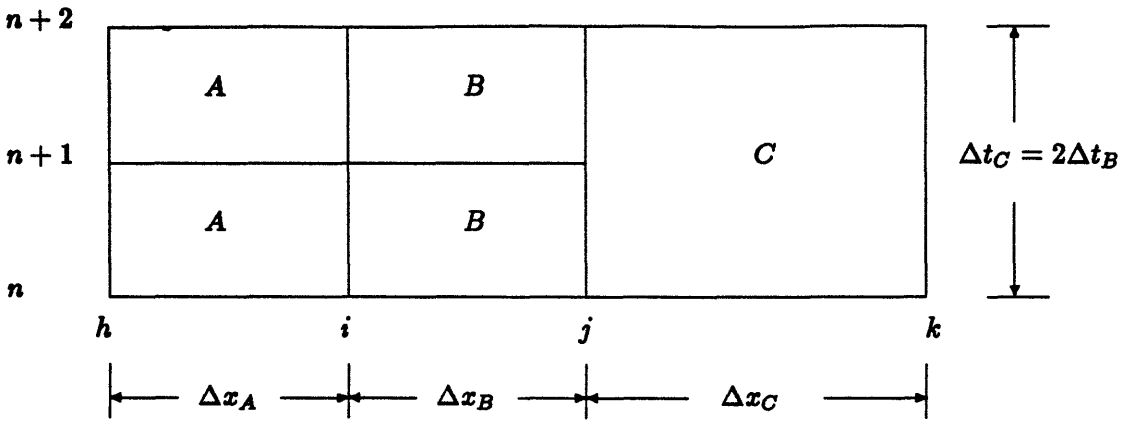

Figure 6.4: Finite volumes adjacent to nodit j.

In the spirit of the predictor corrector scheme, three separate integration passes are proposed in order to advance to time level $(n+2)$. Reference can be made to Figure (6.5) which shows the situation graphically for node j. On the first pass all cells to the left of node j are integrated using time-step Δt_B and change contributions based on level (n) are determined for each cell. After all nodes are updated, those to the left of node j advance to time level $(n+1)$ whereas node j advances to a time level $(*)$, as given by Equation (6.18) with Δt replaced by Δt_B. Obviously data stored in each change contribution variable must be set equal to zero after each updating. The state $(n+1)$ at node i, after updating, is defined in the usual manner by

$$U_i^{n+1} = U_i^n + \delta U_{iA}^{n,n} + \delta U_{iB}^{n,n}. \tag{6.22}$$

On the second pass all cells to the right of node j are integrated using a time-step $\Delta t_C = 2\Delta t_B$ and change contributions for each cell (except C) are determined based upon level (n). The subsequent updating advances all nodes to the right of node j to time level $(n+2)$, whereas node j advances to level $(\dagger)$ given by

$$U_j^{\dagger} = U_j^* + \delta U_{jC}^{*,n}. \tag{6.23}$$

Here the change for cell C is based upon level $(*)$ for node j and level (n) for node k.

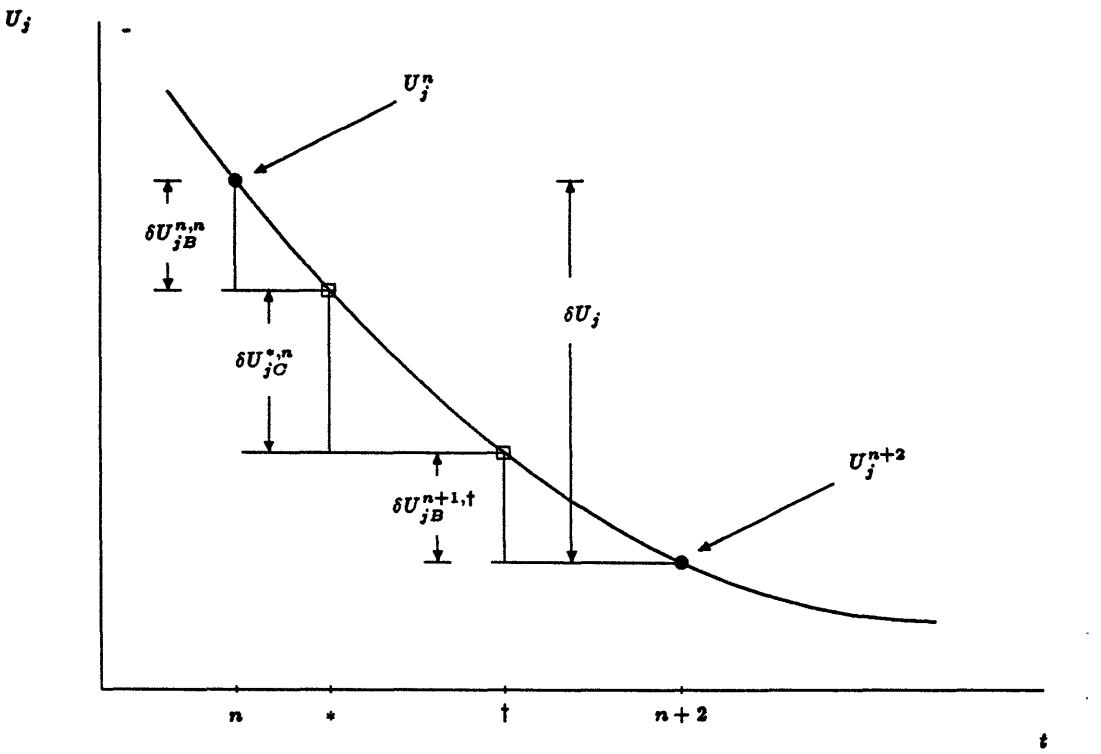

Figure 6.5: Graphical representation for temporal embedding.

The state at node k, after updating, is given by

$$U_k^{n+2} = U_k^n + \delta U_{kC}^{*,n} + \delta U_{kD}^{n,n} \tag{6.24}$$

where D is the cell to the right of node k in Figure (6.4). On the final integration pass all cells to the left of node j are integrated again using the time-step Δt_B and change contributions are determined based upon level $(n+1)$, except for cell B which is based upon level $(n+1)$ for node i and level $(\dagger)$ for node j. The subsequent updating advances all nodes in the computational domain to time level $(n+2)$ with the state at node j given by

$$U_j^{n+2} = U_j^{\dagger} + \delta U_{jB}^{n+1,\dagger}. \tag{6.25}$$

The *exact* conservation property at node j for this multi-step approach dictates

$$F_j(B_1) + F_j(B_2) + 2F_j(C) = 0. \tag{6.26}$$

This is satisfied only to $O(\Delta t_B)$ since

$$F_j(B_1) = -F_j^n \quad , \qquad F_j(C) = +F_j^* \quad , \qquad F_j(B_2) = -F_j^\dagger .$$

Exact conservation can be maintained if the nodit j is recognized to be a temporal interface and the fluxes are frozen as

$$F_j(B_1) = F_j(B_2) = -2F_j(C) = -F_j^n . \tag{6.27}$$

However, this is not done in the developed code, since the implied logic to handle temporal interfaces would undoubtedly be very complex. It is also observed that with this treatment exact conservation property is maintained at nodes j and k and the problem is brushed aside to approximate conservation at node i. Furthermore the generalization of the frozen flux concept for larger time-stride units becomes more complicated, even in one spatial dimension, and the utilization of frozen flux values may hinder the proper propagation of information when the feature within the resolved regions move and influence the nearby regions. The *exact* conservation property was compromised in favor of simplicity in updating and using the latest available information for the nodes.

For the example discussed here the *time-stride* consists of two time-steps for cells to the left of node j and one time-step for the cells to the right. Since each node of a cell is updated after each integration pass, and the flux, etc., are recomputed, the state at a nodit *during* a time-stride at intermediate time-levels is not available; however, on completion of a time-stride the state for all the nodes arrives at the same time level. The use of latest available information means that all the data corresponding to states during a time-stride need not be saved or stored. This also means that that the concept of time-stride can be extended to include larger number of time-steps as will be shown in the latter part of this chapter.

As a final note to this section it is appropriate to point out that it would be misleading to conclude that for all computations of frozen flows involving a uniform spatial grid a global minimum time-step is the appropriate one. For example, for a shock moving in a 1-D stream-tube at a Mach number of 6, the allowable cell time-steps can vary by about a factor of 8 on either side of the shock for a constant CFL number.

Hence temporal adaptation could be useful even when uniform spatial grids are used for non-reacting flows. The utility of temporal adaptation increases further when grids are spatially adapted for transient frozen situations and it is especially attractive for processes involving disparate time-scales which may be coupled with spatial resolution.

6.4 Illustrative Example

As an illustrative example for temporal embedding, consider the following scalar model for $U(x,t)$ with $x \in [0,1]$ and $t \geq 0$

$$\frac{\partial U}{\partial t} + \frac{\partial U}{\partial x} = 0 \tag{6.28}$$

with the initial condition

$$U(x,0) = \mathrm{e}^{2x} \tag{6.29}$$

and the boundary conditions

$$\begin{aligned} U(0,t) &= \mathrm{e}^{-2t} \\ U(1,t) &= \mathrm{e}^{2-2t}. \end{aligned} \tag{6.30}$$

The exact solution of this model is

$$U(x,t) = \mathrm{e}^{2(x-t)}. \tag{6.31}$$

Let us consider two cells B and C with three nodes at $x_1 = 0$, $x_2 = \frac{1}{2}$, $x_3 = 1$ and suppose that the differential equation is integrated numerically to $t = 0.1$. The distribution formulae for this model are

$$\begin{aligned} \delta U_{2B} &= \tfrac{1}{2}\left(1 + \tfrac{\Delta t_B}{\Delta x_B}\right)(U_1 - U_2)\,\tfrac{\Delta t_B}{\Delta x_B} = (1 + 2\Delta t_B)(U_1 - U_2)\,\Delta t_B \\ \delta U_{2C} &= \tfrac{1}{2}\left(1 - \tfrac{\Delta t_C}{\Delta x_C}\right)(U_2 - U_3)\,\tfrac{\Delta t_C}{\Delta x_C} = (1 - 2\Delta t_C)(U_2 - U_3)\,\Delta t_C. \end{aligned} \tag{6.32}$$

The states at nodes $x_1 = 0$ and $x_3 = 1$ are determined by the boundary conditions. Suppose also that the minimum time-step is $\Delta t = 0.05$. The computations for the single step integration scheme would require determination of δU_{2B} and δU_{2C} for two times to update to $t = 0.10$ as shown in Table (6.2). For clarity the changes are shown

	Time	U_1	δU_{2B}	U_2	δU_{2C}	U_3
Start	$t = 0.00$	1.00000		2.71828		7.38906
Changes	$\Delta t = 0.05$		-0.09451		-0.21018	
Update	$t = 0.05$	0.90484		2.41359		6.68589
Changes	$\Delta t = 0.05$		-0.08298		-0.19225	
Update	$t = 0.10$	0.81873		2.13836		6.04965
Exact	$t = 0.10$	0.81873		2.22554		6.04965

Table 6.2: Single step integration based upon $\Delta t = 0.05$.

on separate rows to indicate that they have been based on the states as listed in the previous rows.

Since $|\delta U_{2C}|/|\delta U_{2B}| \approx 2$ the cell time-step for cell B can be increased by a factor of two to make the change contributions at the middle node comparable to the two adjacent cells. This means that we can choose $\Delta t_C = 0.05$ and $\Delta t_B = 0.10$. The computations for this multi-step integration scheme are shown in Table (6.3). As in the previous case two integration steps are needed for cell C but only for cell B. The changes are again shown on separate rows indicating their evaluations based upon the states in the previous rows. The single step scheme involves four change evaluations, two update operations for middle nodes, and four boundary condition calculations, compared to three operations of each type for the multi-step scheme. The latter loses only on the count of update operations for the middle node. However, that operation is

itself very inexpensive, and the multi-step scheme is definitely superior with respect to computational efforts involved in the single step approach. Assuming that the CPU time for the boundary condition evaluation is comparable to that for change computations, the multi-step scheme consumes only 75% as much CPU time compared to the single-step scheme to update the solution to the same time-level.

	Time	U_1	δU_{2B}	U_2	δU_{2C}	U_3
Start	$t = 0.00$	1.00000		2.71828		7.38906
Change	$\Delta t_C = 0.05$				-0.21018	
Update				2.50810		6.68589
Change	$\Delta t_B = 0.10$		-0.18097			
Update		0.81873		2.32713		
Change	$\Delta t_C = 0.05$				-0.19614	
Update	$t = 0.10$	0.81873		2.13099		6.04965

Table 6.3: Multi-Step integration based upon $\Delta t_C = 0.05$ and $\Delta t_B = 0.10$.

As is evident from the two tables, the final result is not identical for the two schemes, but the difference is less than 1%. Both underpredict the value at the middle node by about 4% compared to the exact solution. Decreasing the time-steps does very little to improve the comparison with exact solution. For example if the time-steps are reduced by an order of magnitude the solution at the middle node at $t = 0.10$ is found to be 2.1391 and 2.1383 for the single and multi-step schemes respectively, which represents a

difference of less than 0.1%. This is due to the fact that temporal accuracy is inherently related to the spatial accuracy, and more accurate results only can be obtained by considering finer resolution in both space and time simultaneously.

The results of this illustrative example appear to justify the usefulness of the multi-step scheme in limiting computational resources and efforts while maintaining reasonable temporal accuracy. If time-strides comprised of more than two time-steps can be achieved, further savings in computational efforts can be realized. In fact simultaneous adaptation in both space and time would then yield orders of magnitude faster computations.

6.5 Generalization for Larger Time-strides

6.5.1 One Spatial Dimension

Consider Figure (6.6a) which shows an example of the assignment of cell time-steps as the minimum of both resolution and CFL restrictions before any readjustments. The cell time-steps can be reassigned as multiples (of power of 2) of a global minimum time-step, Δt_{min}, as shown in Figure (6.6b), so that an integral number of integration passes can be completed for cells with the same time-steps. For this example the *size* of the time-stride is $m = 2$. A general procedure for the assignment of the individual steps in a time-stride can be evaluated by the following simple approach.

Based upon cell time-steps given by Equation (6.6) evaluate global minimum and maximum Δt over the entire domain; the *size* m of the time-stride may then be assigned such that

$$2^m \leq \min\{\frac{\Delta t_{max}}{\Delta t_{min}}, 2^M\} < 2^{m+1}. \tag{6.33}$$

Note that m is the *current maximum allowable temporal level* for domain cells and is constrained to be less than or equal to a prescribed maximum level, M. Such a constraint on temporal levels is necessary in order to avoid very long time-stride units which may cause spillage of the feature being resolved from the spatially embedded

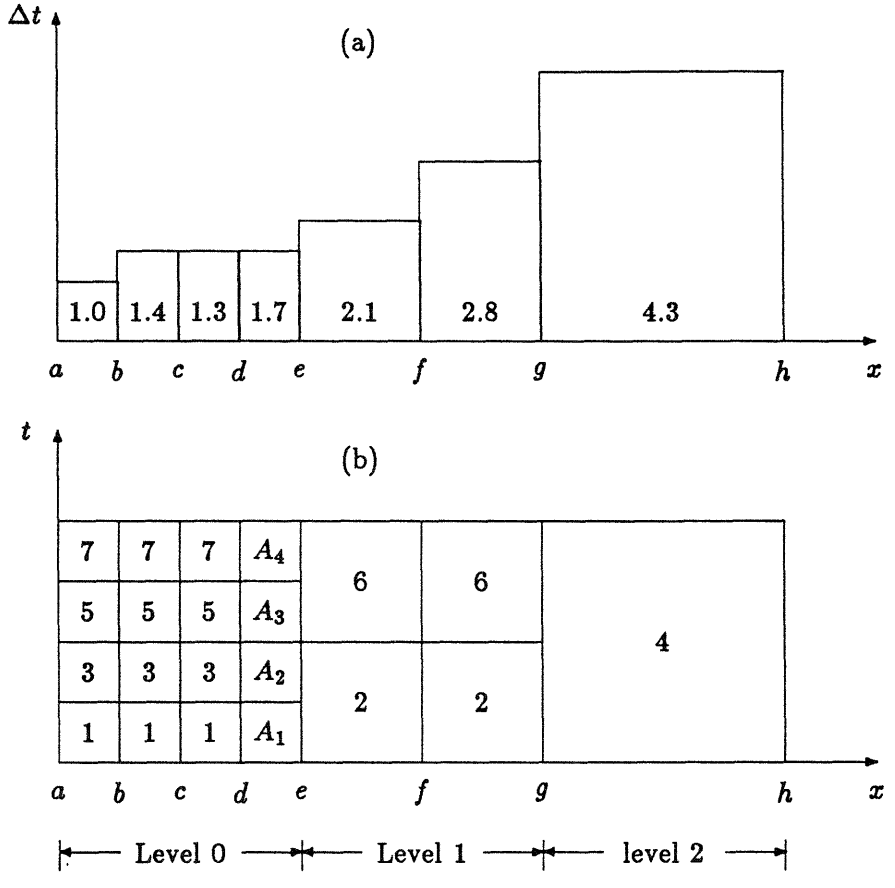

Figure 6.6: Cell time-steps: (a) initial assignment; (b) assignment for temporal adaptation.

region [106]. This phenomenon will be referred to as *temporal level stiffness.* A flow feature generally implies an associated characteristic speed, *e.g.* that for a shock or reaction, and the spatially embedded region must be sufficiently large to ensure that the feature will remain within the embedded region during the subsequent time-stride. The size of a time-stride depends upon the value of M. If a large value for the maximum allowable temporal level m is used then the spatially embedded portion of the grids must be enlarged to avoid departure of the feature from this region. Though temporal adaptation involving long time-strides helps to reduce CPU time, the calculation on increased number of extension nodes can be expensive and a balance between the two competing effects is necessary. To avoid such temporal level stiffness one must not use an extremely large sized time-stride; the current maximum that has been used in this study is $M = 10$.

Actual time-steps for a given cell C are re-assigned according to

$$\Delta t_C^{new} = 2^n \Delta t_{min} \tag{6.34}$$

where the level $n \leq m$ is given by

$$2^n \leq \min\{\frac{\Delta t_C}{\Delta t_{min}}, 2^m\} < 2^{n+1}. \tag{6.35}$$

The total number of time-strides for level n cells is 2^{m-n}.

Another facet of temporal level stiffness is that the time-steps can vary appreciably for contiguous cells. This is improbable for frozen flows because a division into four sub-cells reduces the time-step by a factor of 2 and hence this facet of temporal level stiffness can be avoided by controlling the difference of spatial level embedding between contiguous cells. For reacting cases the source terms can vary appreciably between contiguous cells and hence can cause a corresponding variance of time-steps. To avoid such occurrances the cell time-step is restricted to be at most 4 times the minimum time-step of the surrounding cells.

On completing all readjustments, cells with the same temporal levels are grouped together for subsequent integration. Thus cells with time-steps Δt_{min} are in group level

0, those with $2\Delta t_{min}$ are in group level 1, and so on. The total number of time-steps needed for cells in level n to advance to the next time-stride is 2^{m-n}.

The order in which the integration takes place over the cells is of special importance [81,106]. Successive integrations over the same cell in passing from one iso-temporal surface to the next will produce a degraded solution since information from neighboring cells is not allowed to propagate. For example, if the level 0 cells labelled A_i in Figure (6.6b) are integrated four times consecutively use is made only of information based on the two nodes d and e. This is correct for A_1, but for A_4 additional account must somehow be taken of the nearby nodes. If for seven integration passes, we integrate level 0 cells on pass 1, level 1 cells on pass 2, level 0 cells on pass 3, level 2 cells on pass 4 and so on as indicated by the numbers in Figure (6.6b) by the time A_4 will be integrated the nodes d and e will have accumulated effects from nodes a through g, provided that after each integration pass the cells at a particular level have been updated and the flux, source terms and Jacobians recomputed. This represents yet another facet of temporal level stiffness. In general, a cycle of P_T integration passes completes a time-stride unit, the total number being

$$P_T = 2^{m+1} - 1. \tag{6.36}$$

On pass $P \in [1, P_T]$ cells with temporal level n are integrated if

$$\frac{P - 2^n}{2^{n+1}} = \text{integer}. \tag{6.37}$$

6.5.2 Two Spatial Dimensions

The ideas developed for one spatial dimension hold for multi-dimensions as well. As an example consider the time-stride in Figure (6.7) with $M = 2$ as the prescribed maximum time-level of cells. Also suppose that the time-step variation is such that the current maximum allowable temporal level of $m = 2$ is possible, and therefore $P_T = 7$. For clarity of view a slice has been removed from the figure. The dots on the top surfaces of each cell indicate the time-step as a multiple of the global minimum time-step. Hence, cells with one, two, four dots are at temporal levels $n = 0, 1, 2$, *etc.* The

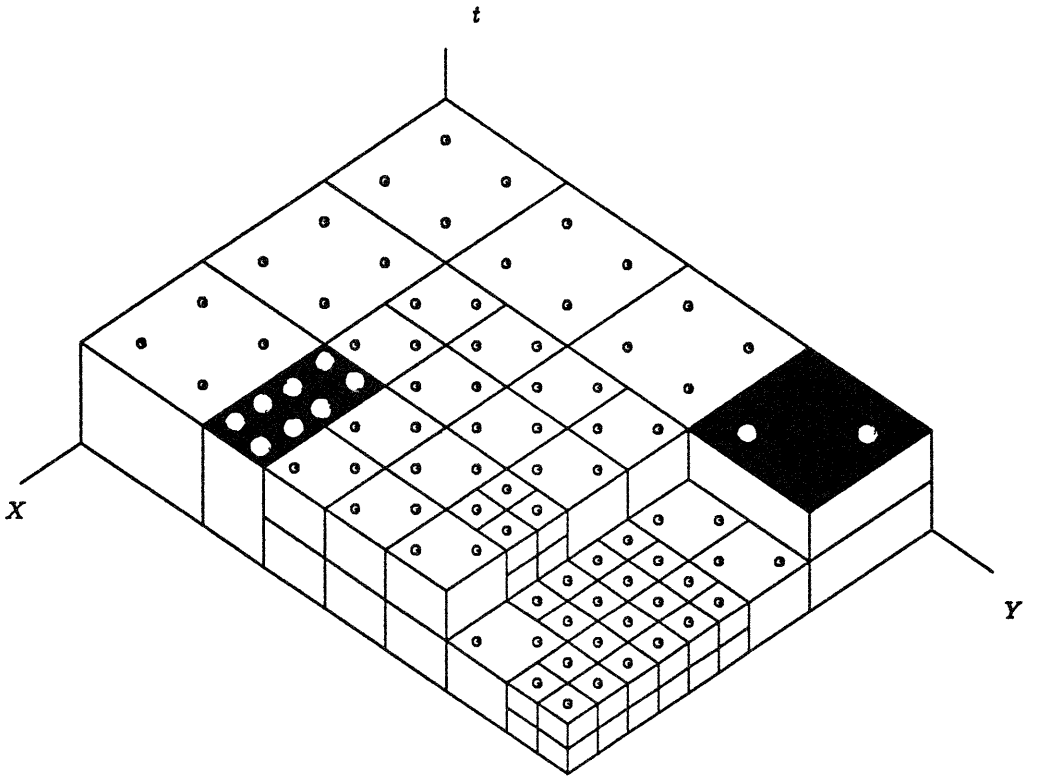

Figure 6.7: Time-stride with $m = 2$.

correspondence between the integration passes and temporal cell levels still follows from Equation (6.37):

P	1	2	3	4	5	6	7
n	0	1	0	2	0	1	0

The chemical source terms may alter the time-step distributions in such a way as to create cells with spatial resolution in the absence of temporal resolution (cells with 4 dots and shaded top surfaces) and temporal resolution in the absence of spatial resolution (cells with 2 dots and shaded top surfaces). Such complications do not exist for frozen flow computations. The makeup of the time-stride changes with the movement of the flow features being resolved, and a different number of levels may exist for consecutive time-strides. The generation of fresh time-strides depends upon the velocity of the features; hence for fast moving features time-strides should be renewed after each spatial adaptation operation and *vice versa*.

6.5.3 Summary

Temporal and spatial adaptation procedures are inherently different and are applied separately. When the spatial adaptation process is carried out it is at a current time level, at all spatial locations, and done infrequently relative to the number of temporal adaptations. Nevertheless, the frequency of spatial adaptations does depend upon the time rate of change of the flow feature being resolved. On the other hand, temporal adaptation is repeated after each time-stride at all spatial locations and must anticipate subsequent changes in the the flow field as the features move. The four steps needed for completing a temporal grid adjustment are :

1. a determination of an allowable Δt for each cell,
2. reassignment of Δt to be multiples (of power of 2) of global minimum time-step, Δt_{min},

3. further reassignment of Δt distributions such that adjacent cells vary at most by a factor of 2 in 1-D and a factor of 4 in 2-D, and

4. determination of a proper integration sequence over the cell domain.

Nodes at the boundary of cells with different time-steps, *nodits* , are not necessarily the same as middle nodes for spatial interfaces. No special formulation is needed at nodits and in order to render the actual spatial location of any temporal level cell irrelevant, a data base must be constructed so as to store cells at same temporal level together. There is no such restriction for spatial adaptation pointers. The choice of such a data base allows the calculations for each pass to be performed in parallel; the data dependencies occur only at nodits between various passes for a given integration sequence. However, the integration order does not strictly have to follow the aforementioned sequence (Eq. 6.37) at nodits, and such data dependencies will cause only slight variations between parallel and non-parallel calculations. Since all nodes of a cell are updated after each integration pass, and the state vector, *etc.*, are recomputed, the state at a nodit *during* a time-stride is not correct at intermediate time levels; however, on completion of a time-stride the state does correspond to a correct time.

Chapter 7

Initial and Boundary Conditions

The solution of a reactive system is determined by the initial condition, the set of conservation and constitutive relations and the boundary conditions. The initial condition is a spatial distribution of the state vectors when the computation is initiated, usually at "zero" time. Boundary conditions describe the exchange of mass, momenta, energy and species between the system and the external universe through its boundaries. These conditions can have both physical and numerical implications, and each can influence the numerical solution in a different manner. This chapter describes the initial and boundary conditions, both from a numerical and physical point of view. The boundary conditions are discussed only for two spatial dimensions. Following some introductory remarks the initial conditions are discussed in Sections (7.2) and (7.3). A characteristic analysis for the purpose of applying numerical boundary conditions at inflow and outflow is discussed in Section (7.4). The boundary conditions for solid wall, inflow/outflow are discussed in Section (7.5).

7.1 Introduction

To begin the solution of the finite volume equations in time, it is necessary to specify a set of initial conditions for each node in the computational domain. These include specification of geometry (independent variables) and state vectors (dependent variables) at initial time. Values are also required for thermophysical data and other input parameters. The thermophysical data includes information like number of species and reactions, stoichiometric coefficients, constants in rate coefficient expressions, specific heats, heats of formation, molecular masses, and threshold temperature for by-passing

source term computations. Other input data pertains to adaptation parameters such as spatial and temporal adaptation criteria variables, number of cells to be extended after each adaptation cycle, predefined threshold values, *etc.* Quantities such as flux vectors, source vector, *etc.* can be initialized by direct computations involving the state vectors and the thermophysical data.

The specification of initial conditions and input parameters is generally regarded to be easier than imposing boundary conditions because these are input just at the beginning of the calculation for a fresh start rather than executed as constraints after each time-stride. However, for reacting flows, the initialization of a whole slew of thermophysical data and consistencies in the values of state vectors which depend upon the reaction system can be some-what time-consuming and prone to errors.

Uniform initial conditions are specified for some cases in this thesis through the input data at the upstream boundary. This is quite straight-forward and nothing more need be stated about its implementation. Two other kinds of initial conditions have been considered for either a perfect or Lighthill gas. These are

- conditions across a diaphragm in a shock tube
- conditions across a moving shock.

These will be discussed in Sections (7.2) and (7.3) respectively.

Boundary conditions frequently involve special constructions which are applied at the boundaries of a computational domain. The term *physical* boundary condition is used here to describe assumed flow conditions along the boundaries of a domain. In addition there are *numerical* boundary conditions which impose additional restraints to close the system of discrete equations. Special numerical formulations are needed at the boundaries because some of the cells adjacent to the boundary nodes are non-existent, and hence integration procedures cannot be applied at these nodes in the same manner as at interior nodes. In particular, the construction of boundary conditions should be simple, mathematically tractable and physically meaningful. Several different types of

boundary conditions such as those for inflow, outflow, free surfaces, fluid interfaces and rigid walls are required for computing solutions and each of these requires a different mathematical and numerical treatment. The types of boundary conditions considered in this chapter are

- free slip rigid walls
- prescribed input (supersonic inlet)
- continuitive output (supersonic exit)
- subsonic inlet
- subsonic exit

The inlet/exit boundary conditions may be applied through a characteristic analysis as discussed in Section (7.4).

7.2 Initial Conditions for Shock Tubes

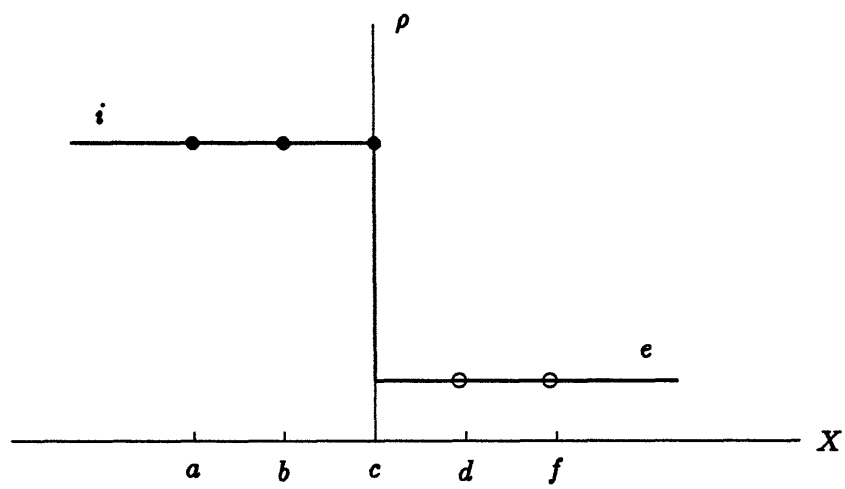

Figure 7.1: Initial distribution of density across a shock tube diaphragm.

Shock tube property distributions constitute step functions in terms of the state values at $t = 0$ as shown in Figure (7.1) for a typical density distribution. Location c denotes the contact surface and stations i, e indicate inlet, exit of the computational domain which are respectively the high, low pressure sides. Hence when the diaphragm is shattered, the contact surface and shock discontinuity move from left to right. The assignment of state values at location c may be regarded as that of inlet, exit or simply the mean value, but for the sake of discussion here this mesh point is regarded to be a part of the inlet condition. Clearly it is helpful to generate grids which align with the contact surface at the initial time. The symbols in the figure indicate nodes of a computational domain; solid circles indicate the high pressure side and the empty circles indicate the low pressure side.

Initial ratios for temperature, T_e/T_i and density, ρ_e/ρ_i are used as parameter values for this case and the reference values are those at the inlet for the sake of normalization (*i.e.*, $T_i = p_i = \rho_i = 1$ in non-dimensional units). These values are convenient because then the degree of dissociation can be directly computed for either frozen or equilibrium flows. The pressure ratio is computed from

$$\frac{p_e}{p_i} = \frac{\rho_e}{\rho_i}\frac{T_e}{T_i}\frac{1+Y_e}{1+Y_i} \tag{7.1}$$

where Y indicates the degree of dissociation or the mass fraction of dissociated atoms for the assumed Lighthill model. For frozen flow $Y_e = Y_i$ and the ratio reduces to that for ideal gases when the characteristic temperature θ_d is regarded as zero. Note that the non-dimensional thermal equation of state for both perfect gases and frozen Lighthill model (irrespective of θ_d) is

$$p = \rho T$$

whereas the caloric equation of state for perfect gases is

$$\frac{\epsilon}{\rho} = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}V^2$$

and it has the same form as the Lighthill model when $\theta_d = 0$ and $\gamma = (4+Y)/3$ (See Section 2.7). For frozen flows the reaction parameter Φ is zero and any constant value for the degree of dissociation can be used; for comparison with perfect gases this value can be chosen so as to correspond to a given value of ratio of specific heats. For non-zero

values of the reaction parameter, initial values of the degree of dissociation across the contact surface can be specified independently. However it is reasonable to assume that if the fluid had been present in the two sections for a sufficiently long time period then equilibrium values of degree of dissociation do exist. These values are given by

$$Y_k = \frac{\sqrt{1 + 4\, e^{\theta_d/T_k}\, \rho_k/\rho_d} - 1}{2\, e^{\theta_d/T_k}\, \rho_k/\rho_d}. \tag{7.2}$$

The velocity components across the contact surface are initially chosen to be zero and the energy term is given by the caloric equation of state (Eq. 2.33 or 2.98).

For a frozen flow, the shock speed M_s is given by the following implicit relation [77]

$$\frac{p_e}{p_i} = \frac{\gamma_e + 1}{2\gamma_e M_s^2 - (\gamma_e - 1)} \left[1 - \frac{a_e}{a_i}\left(\frac{\gamma_i - 1}{\gamma_e + 1}\right)\left(M_s - \frac{1}{M_s}\right)\right]^{\frac{2\gamma_i}{\gamma_i - 1}} \tag{7.3}$$

where the shock Mach number M_s is defined to be the shock speed divided by the frozen speed of sound in the downstream section e. Thus if a shock of a given strength is desired, this relation can be used to determine the overall pressure ratio and other values can be evaluated therefrom.

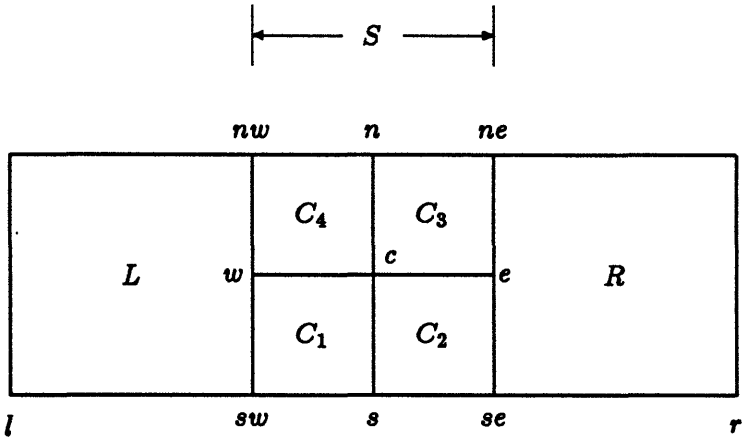

Figure 7.2: Cell division after a single adaptive pass.

The initial condition is generally specified on a coarse grid, and a direct integration of the system of equations from this grid would degrade the subsequent solution. This is because the newly created nodes between the nodes c and d in Figure (7.1) would then

have values assigned to them as given by linear interpolation. Hence the grid should be pre-embedded prior to an execution of the integration procedure. The number of calls to the pre-embedding procedure would equal the maximum spatial level of the cells desired. The pre-embedding procedure is explained here for a 2-D grid when a step function input is involved. The pertinent spatial grid after one adaptive pass is shown in Figure (7.2). For simplicity the left L and right R cells are shown as undivided, but this is not a necessity. The adaptive procedure assigns values at the newly created nodes c, s, e, n, w based upon the interpolated values from the nearby nodes and this may not be consistent with the initial step function. The pre-embedding cycle follows this, examines the newly created nodes and reassigns the values at these nodes. Since the initial condition is assumed to be 1-D in nature, same conditions are applied for nodes lying on a vertical grid line. The procedure is accomplished as follows:

1. Save the total number of cells N_p prior to a spatial adaptive cycle.
2. Invoke the usual spatial adaptation procedure while not allowing grid fusion. This increments the total number of cells to N_c.
3. Examine all cell numbers between N_p and N_c in steps of 4 (since four cells form a bigger unit). Suppose the cell under consideration in Figure (7.2) is C_1; then
   (a) Find Supercell S of this cell by cell-to-node array.
   (b) Locate all the nodes of cell S by the cell-to-node array, and locate the cells R, L by the node-to-cell array and their nodes in a similar manner.
   (c) If the state at nodes *sw* and *se* nodes is identical, then set states at all newly created nodes of cell S equal to that of node *sw*;
   otherwise, if the state at *se* node is identical to that of node *r*, then set the states at nodes c, s, e, n equal to that of node *r* and the state of node *w* equal to that of *sw*;
   otherwise, if the state at *sw* node is identical to that of node *l*, then set the states at nodes c, s, n, w equal to that of node *l* and the state of node *e* equal to that of *se*.
   (d) Proceed to examine the next cell in the list (go to 3a).

4. If the desired level of spatial resolution is not yet achieved, repeat the entire process again (go to 1).

7.3 Initial Conditions for Moving Shocks

7.3.1 Frozen Flow

Consider a single shock propagating initially along a straight channel as shown in the schematic diagram of Fig. (7.3). For a frozen flow the relaxation distance behind the normal shock becomes infinitely large and station f can be regarded to be at the same state as that of station i. The conditions across the moving shock with shock Mach number M_f and ratio of specific heats γ are given by [77]

$$\begin{aligned} p_i/p_e &= \frac{2\gamma}{\gamma+1}M_f^2 - \Gamma \\ \rho_i/\rho_e &= \frac{\Gamma + p_i/p_e}{1+\Gamma p_i/p_e} \\ u_e &= 0 \\ u_i &= \frac{2}{\gamma+1}a_e(M_f - 1/M_f) \end{aligned} \tag{7.4}$$

where $\Gamma = (\gamma-1)/(\gamma+1)$ and $a_e^2 = \gamma p_e/\rho_e$.

Note that unlike the previous type of initial condition, there is a non-zero mass influx at the inlet boundary which is responsible for the forward motion of the shock. However, the two flow conditions should yield similar results if the shock Mach number is the same and the region of interest is far away from the starting position of the contact surface, *i.e.*, the location of unruptured diaphragm.

Since for frozen flow the initial condition is a step function, the same pre-embedding technique can be used as in the shock tube case.

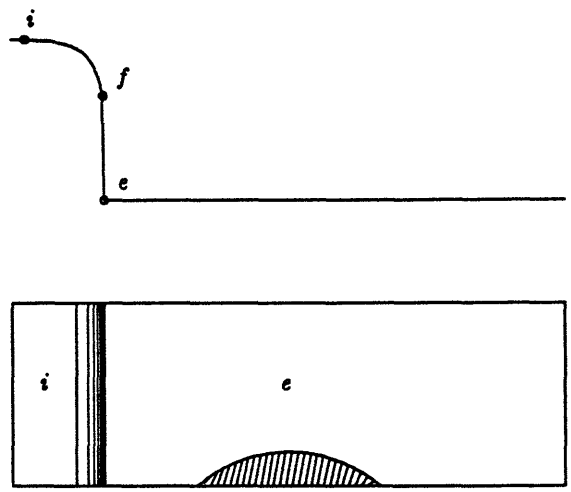

Figure 7.3: Initial density variation for a relaxation behind a shock.

7.3.2 Lighthill Gas

In contrast to the frozen case, the initial conditions for a partially dispersed shock involves a jump (station e to f), followed by a relaxation tail (station f to i) which is characterized by a gradual adjustment to equilibrium. The overall change between stations e and i is given by the equilibrium shock relations [136] and after some algebra can be written as

$$\begin{aligned}
\frac{\rho_i T_i}{\rho_e T_e}\left(\frac{1+Y_i}{1+Y_e}\right) &= 1+M_f^2\left(\frac{4+Y_e}{3}\right)\left(1-\frac{\rho_e}{\rho_i}\right) \\
(4+Y_i)T_i+Y_i\theta_d &= (4+Y_e)T_e+Y_e\theta_d+ \\
&\quad M_f^2\left(\frac{4+Y_e}{6}\right)(1+Y_e)\left[1-\frac{\rho_e^2}{\rho_i^2}\right] \qquad (7.5)\\
(1-Y_k)e^{-\theta_d/T_k} &= \frac{\rho_k}{\rho_d}Y_k^2, \qquad k=i,e.
\end{aligned}$$

Here Y_k represents the mass fraction of dissociated atoms at station k and M_f represents the *frozen* shock Mach number which is given by

$$M_f^2 = \frac{u_f^2}{\gamma_e p_e/\rho_e} \quad \text{with} \quad \gamma_e = \frac{4+Y_e}{3}. \qquad (7.6)$$

The O.D.E. for the mass fraction in the relaxation zone is

$$\frac{dY}{dx} = \Phi T^\eta \frac{\rho}{w}\left[(1-Y)e^{-\theta_d/T}-\frac{\rho}{\rho_d}Y^2\right] \qquad (7.7)$$

where w is the fluid velocity in a frame of reference attached to the frontal shock. The integration is started from the initial frontal shock location ($x = x_f$ at $t = 0$) with $Y = Y_f = Y_e$. The other quantities at station f are given by the *frozen* shock relations between stations f and e, that is,

$$\begin{aligned} w_e^2 &= M_f^2 \frac{\gamma_e p_e}{\rho_e} \\ \rho_f/\rho_e &= \frac{\gamma_e + 1}{2 + (\gamma_e - 1) M_f^2} \\ T_f/T_e &= 1 + \frac{\gamma_e - 1}{2} M_f^2 \left(1 - \rho_e^2/\rho_f^2\right). \end{aligned} \tag{7.8}$$

The usual integral conservation relations connecting state f and current value at any place inside the relaxation zone are

$$\begin{aligned} \rho w &= \rho_f w_f &&= \rho_e w_e &&= C_c \\ \rho w^2 + p &= \rho_f w_f^2 + p_f &&= \rho_e w_e^2 + p_e &&= C_m \\ (\epsilon + p)/\rho &= (\epsilon_f + p_f)/\rho_f &&= (\epsilon_e + p_e)/\rho_e &&= C_\epsilon. \end{aligned} \tag{7.9}$$

These relations would imply the following for the relaxation zone

$$\rho = C_c/w \tag{7.10}$$

$$p = C_m - C_c w \tag{7.11}$$

with the velocity in the stationary frame given by

$$w = \frac{B - \sqrt{B^2 - 4AD}}{2A} \tag{7.12}$$

where

$$\begin{aligned} A &= 7 + Y \\ B &= 2(4 + Y) C_m/C_c \\ D &= 2(1 + Y)\left(C_\epsilon - Y\theta_d/2\hat{m}_Z\right). \end{aligned} \tag{7.13}$$

Note that the second term in the last parenthesis is normalized by the reference heat of formation u_r^2 and the corresponding dimensional term is $Y\mathcal{R}\theta_d/2\hat{m}_Z$. Hence all other variables in Equation (7.7) can now be written in terms of Y and as such the equation can be integrated. Once velocities w in the stationary shock frame are known, the velocities u in the moving shock frame follow from

$$u = w_e - w. \tag{7.14}$$

A relaxation length, x_t, is defined as the distance between the shock discontinuity at f and where dissociation reaches 99% of the equilibrium value (*i.e.*, $0.99Y_i$). Hence x_t can be used as a convenient measure of non-equilibrium between its zero (equilibrium) and infinite (frozen) limits.

A different pre-embedding approach is needed in this case. When Equation (7.7) is integrated to yield property variations a separate file is written for all state vector components as function of distance x at all the step-sizes which are assumed to be reasonably fine. When pre-embedding is desired for this case and allocation of state vector at a node like s (Fig. 7.2) is under consideration, this file is scanned to locate the appropriate x-locations within which this node lies. These locations are denoted by x_i, x_{i+1} in Figure (7.4). A linear interpolation is then used between these two locations for determination of state vector at node s.

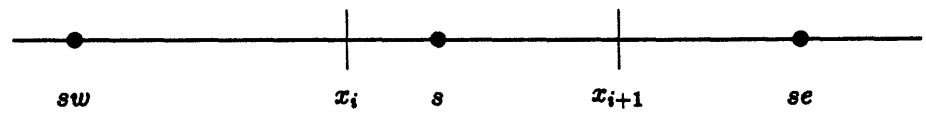

Figure 7.4: Allocation of state vector at s, based upon linear interpolation between x_i and x_{i+1} of a previous integration procedure for an O.D.E..

7.4 Characteristic Analysis

The inviscid governing equations possess a set of real eigenvalues and a set of linearly independent eigenvectors can be determined corresponding to each of these, as detailed in Appendix A. The eigenvalues describe the ***characteristic directions*** along which the variations of ***characteristic variables*** is known. These variables remain constant for frozen flows and are termed as Riemann invariants.

The sign of each eigenvalue determines the direction of propagation of characteristic variables and has implications that pertain to physical boundary conditions. The literature involving characteristic propagation for frozen flows is very rich; some of the readily available sources of information on the subject are Courant and Friedrichs [30], Friedrichs [53], Ferri [51], Meyer [90], and Shapiro [118]. The application of the theory to yield well-posed numerical boundary conditions is discussed by Chakravarthy [27]. Aspects of the general theory that apply to reacting flows are discused by Vincenti and Kruger [136] and Sedney [116]. A literature search on relaxing flows did not reveal any a source reference that treats the well-posedness of numerical boundary conditions in a consistent manner so far as the propagation along characteristics is concerned.

A feature of non-equilibrium flows, that had stirred controversy in earlier studies during the fifties, pertains to the proper choice of sound speed, especially in the limiting case of nearly equilibrium flows. A proper choice of characteristic directions is crucial to a successful numerical calculation. The theory of characteristics shows that the proper directions correspond to a frozen speed of sound and the flow velocity. In fact there is no apparent reason for not using the local frozen characteristics in a calculation of equilibrium flow [116]. Another aspect pertains to the multiplicity of the characteristic eigenvalues, due to the similar nature of species equations and hence a number of characteristic directions, each with a different behavior, must be treated.

Since the governing equations are *quasi-linear*, they can be written as

$$\frac{\partial U}{\partial t} + F_U \frac{\partial U}{\partial x} + G_U \frac{\partial U}{\partial y} = W. \tag{7.15}$$

Note that this linearization does not represent an approximation if the state vectors are written in terms of primitive variables and this system is equivalent to Equations (2.48). Denoting the left eigenvector matrix of the first flux Jacobian by L, the governing equations can be further written as

$$L\frac{\partial U}{\partial t} + \Lambda L \frac{\partial U}{\partial x} + L G_U \frac{\partial U}{\partial y} = LW \tag{7.16}$$

where Λ is a diagonal matrix with entries equal to the eigenvalues of F_U. By definition, a left eigenvector satisfies the following identity

$$\Lambda = L F_U L^{-1}. \tag{7.17}$$

The diagonal entries of this matrix, as shown in Appendix A, are given by

$$\text{diag}\Lambda \;=\; [u - a_f, u + a_f, u, \cdots, u] \tag{7.18}$$

The order of the eigenvalues need not be in the above form, but it is consistent with the derivation presented in Appendix A. Note that no approximation has been introduced into Equations (7.16) and as such the system represents coupled equations even when variations along the y-direction are negligible. However, if the left eigenvector is assumed to be locally frozen (constant in both space and time), the system becomes an uncoupled set

$$\frac{\partial Q}{\partial t} + \Lambda \frac{\partial Q}{\partial x} + \frac{\partial R}{\partial y} \;=\; Z \tag{7.19}$$

if the variations along y-direction can be neglected. Here the new variables are given by

$$Q \;=\; LU, \qquad R \;=\; LG_U U \;=\; LG, \qquad Z \;=\; LW. \tag{7.20}$$

Although L changes with U, the changes in L have been regarded as of a higher order compared to those in the state vector. This is analogous to curve fitting in which linear segments are used and local slope of individual segments are regarded constant while the intercept is allowed to vary.

Consider a coordinate system (s, n) along and normal to the streamlines. This coordinate system is generally known as the natural coordinate system [77,90]. Also assume that the streamlines are locally straight (*i.e.*, the infinite local radius of curvature). The velocity components in this system are $(V,0)$ and Equation (7.19) under the special assumptions becomes

$$\frac{\partial q_i}{\partial t} + \lambda_i \frac{\partial q_i}{\partial s} + \delta_{i3} \frac{\partial p}{\partial n} \;=\; z_i\,, \qquad i = 1, \ldots, N_e. \tag{7.21}$$

Here lower case letters are used to denote the components of the corresponding vectors. Note that the entry in the third row and column of L is $L_{33} = 1$ by construction (see Appendix A) and that with the exception of the third row all the other equations have been decoupled. However, in the absence of curvature effects, the normal momentum equation simply reduces to [1]

$$\frac{\partial p}{\partial n} \;=\; 0. \tag{7.22}$$

[1] See, for example, the third row of Eq. 2.48 in which v is replaced by 0 when writting in a locally rotated natural coordinate system.

The vanishing of normal pressure gradient does not constitute an additional assumption, but is a consequence of locally straight streamlines. Hence the decoupled set of equations is now a system of first order quasi-linear partial differential equations, *i.e.*,

$$\frac{\partial q_i}{\partial t} + \lambda_i \frac{\partial q_i}{\partial s} = z_i \,, \qquad i = 1, \ldots, N_e. \tag{7.23}$$

This has a ***characteristic solution***, which is given by [26]

$$\frac{dt}{1} = \frac{ds}{\lambda_i} = \frac{dq_i}{z_i} \,, \qquad i = 1, \ldots, N_e. \tag{7.24}$$

Note that for frozen flow ($z_i = 0$) the ***characteristic variables*** q_i are constants along the characteristic directions $ds = \lambda_i dt$. Although the characteristic variables q_i for the relaxing flows in the simplified model of Equations (7.23) are not constants, their behavior along characteristic directions is known from Equation (7.24). As can be noted from this equations, these directions are along particle paths. Consider an exit boundary location P, as depicted in Figure (7.5), adjacent to two computational cells. The characteristic direction in this figure is regarded as positive, $\lambda_i = u$; in which case the information propagates along a straight line through point P and in the indicated direction. Also shown in the figure is the space-time grid. All values are assumed known at time t_0 and it remains to determine the values after a time-step Δt at node N. Using the eigenvalue, the slope of the path for the corresponding characteristic variable can be determined. The distance Δs is given by Equation (7.24) as

$$\Delta s = \lambda_i \Delta t \tag{7.25}$$

which determines an interior location I and hence interpolated values at this point can be determined. The characteristic variable q_i at node N is then the interpolated value at location I plus the variation as determined by Equation (7.24), *vis-á-vis*

$$q_N = q_I + z_I \Delta t. \tag{7.26}$$

The previous two results become more accurate as the step-sizes become smaller. The point I lies within one of the two cells adjacent to node P if the CFL constraint for time-steps is satisfied. A similar procedure applies for all other characteristic variables. Once all of the components of Q are determined at node N, the state vector value can

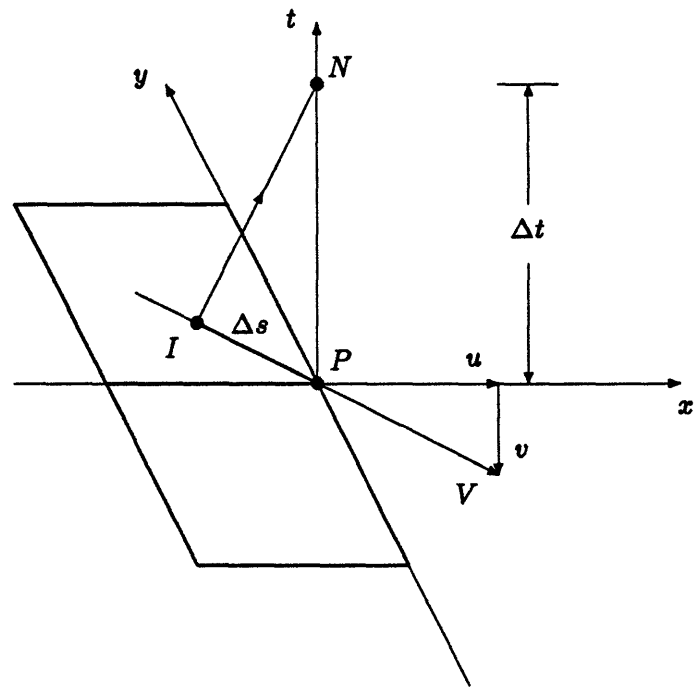

Figure 7.5: Characteristic propagation at an exit boundary along a streamline.

be evaluated by the inverse relation

$$U = L^{-1}Q. \tag{7.27}$$

In what follows this characteristic formulation is applied at inlet and exit boundaries of the computational domain.

7.5 Boundary Conditions

7.5.1 Free Slip Rigid Walls

For inviscid flow the appropriate physical condition on a solid surface is that there be no flow normal to the surface, or equivalently that the flow direction be tangential

to the wall. In mathematical form this condition becomes

$$\mathbf{V} \cdot \hat{n} = 0 \tag{7.28}$$

where $\hat{n}$ is a unit normal vector pointing outward from the surface. At locations where slope discontinuities exist, such as the location along which two flat surfaces intersect, a unique normal direction does not exist and hence this condition cannot be applied rigorously. Consider four such nodes a, b, c, d in Figure (7.6) with the surrounding cells

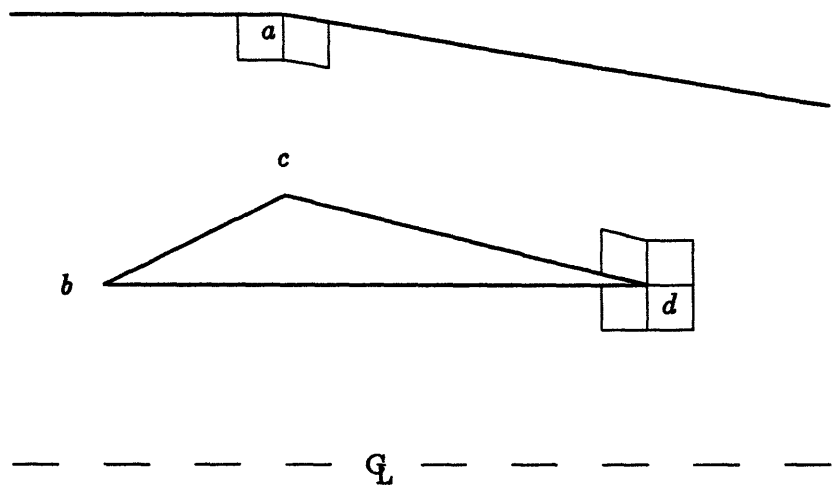

Figure 7.6: Configuration with slope discontinuities.

for nodes a and d. For numerical purposes an average slope may be assumed at nodes a and c whereas nodes b and d may be treated as "interior" nodes since there are four cells surrounding these nodes for small intersecting angles. The latter of these treatments may be questionable if there is not enough resolution surrounding the nodes or if the intersection angle is too large. However, in the current algorithm spatial resolution is expected by virtue of the adaptive technique; hence this treatment may be reasonable for small intersecting angles. Note that free slip rigid wall conditions on the nodes just

upstream of the trailing edge are still applied and hence tangency conditions hold just upstream of the trailing edge. Similarly these conditions are satisfied just downstream of the leading edge.

The application of characteristic theory at a solid wall becomes complicated since the solid wall itself is along a characteristic direction, *i.e.*, a streamline. Therefore an alternative approach is presented here. This treatment is similar to the predictor-corrector approach described by Hall and Salas [61] which involves an image principle.

Consider a node i, Figure (7.7), on a solid wall which makes an angle α with the x-axis. The application of the integration scheme to cells A and/or B yields the change

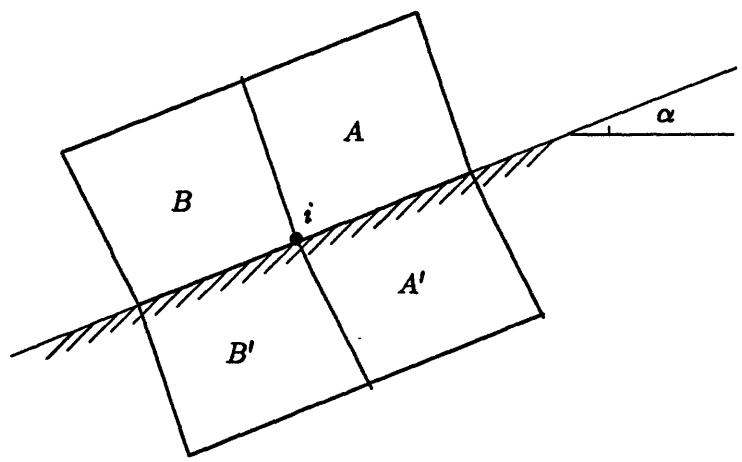

Figure 7.7: Images of cells adjacent to a wall.

at this node in a certain integration pass as

$$\delta U_i = \delta U_{iA} + \delta U_{iB}. \tag{7.29}$$

In fact there are four possibilities for the change at node i, when the integration involves

temporal adaptation. After a certain integration pass (among a total of P_T as discussed in Chapter 6) which involves the integration over cells with a given temporal level, one of the following cases may exist:

- both cells A and B belong to the same temporal level and $\delta U_i = \delta U_{iA} + \delta U_{iB}$
- neither cell A nor cell B exists at the same temporal level and $\delta U_i = 0$
- only cell A belongs to the temporal level and $\delta U_i = \delta U_{iA}$
- only cell B belongs to the temporal level and $\delta U_i = \delta U_{iB}$.

If the boundary conditions are updated at all boundary points after each integration pass, there is no need to discriminate between these individual cases. The boundary conditions are applied at all boundary nodes belonging to cells on a given temporal level, even for nodes which fall in the second category as listed above. This is done in favor of retaining simplistic logic and is not computational expensive since the total number of boundary points is much smaller compared to the total number of nodes in the domain. Further note that if the boundary conditions are applied after each time-stride (instead of after each integration pass) the logic would become very complicated and such treatment may in fact introduce errors which hinders the proper flow of information during intermediate passes. The *predicted change* at node i is taken to be that from cells A, B and their corresponding mirror images A', B' which contribute the same values, *i.e.*,

$$\delta U_i^p \;=\; 2\delta U_i. \tag{7.30}$$

If these values are not corrected, then the wall surface would be a line of symmetry for all variables, including of course the normal component of the velocity. The tangential component of velocity is given by

$$V_t \;=\; u\cos\alpha + v\sin\alpha \tag{7.31}$$

and only this is used to reassign new velocity components along the coordinate directions, *i.e.*,

$$u \;=\; V_t\cos\alpha\,, \qquad v \;=\; V_t\sin\alpha. \tag{7.32}$$

Thus the corrected values for the changes are

$$\begin{aligned} \delta(\rho u)_i^c &= (\rho V_t)^* \cos\alpha - (\rho u)_i^n \quad \text{component 2} \\ \delta(\rho v)_i^c &= (\rho V_t)^* \sin\alpha - (\rho v)_i^n \quad \text{component 3} \\ \delta U_i^c &= \delta U_i^p \quad \text{otherwise} \end{aligned} \tag{7.33}$$

where

$$(\rho V_t)^* = [(\rho u)_i^n + \delta(\rho u)_i^p]\cos\alpha + [(\rho v)_i^n + \delta(\rho v)_i^p]\sin\alpha. \tag{7.34}$$

Dannenhoffer [33] has demonstrated that the doubling of corrections predicted by the standard distribution formulae and a subsequent correction by setting the normal momentum equal to zero yields a correct propagation of changes when the solid wall is aligned with x-axis. Usab [133] has conjectured that this form follows from the fact that the Ni scheme implies a mathematical signal propagation phenomenon from the interior grid points that is analogous to the theory of characteristics.

Several observations are in order here. First note that no extrapolation is involved in the application of solid wall boundary condition. The free slip boundary happens to act as if it was non-catalytic, *i.e.*, $\partial Y/\partial n = 0$, where n is a normal direction to the solid wall. The normal gradients of all dependent variable components, except for velocity vector, are zero, because the predicted and corrected values are the same and these assume that the wall is a symmetry line. The temperature condition at the surface similarly behaves as if it was adiabatic or non-conducting *i.e.*, the condition $\partial\epsilon/\partial n = 0$ implies that the caloric equation of state becomes

$$\rho C_p \frac{\partial T}{\partial n} = \frac{\partial p}{\partial n}$$

for a mixture makeup of components with constant specific heats. Furthermore the thermal equation of state implies that

$$\frac{\partial p}{\partial n} = \rho(C_p - C_v)\frac{\partial T}{\partial n}.$$

These conditions together imply $\partial p/\partial n = \partial T/\partial n = 0$. In fact these conditions hold even when the specific heats are general functions of temperature.

7.5.2 Inflow Boundaries

For supersonic flow all eigenvalues are positive and all the characteristics propagate from the free-stream into the interior of the domain. Thus all characteristic variables q_i can be specified as function of time and distance along the boundary. Alternatively the components of the state vector can be assigned arbitrarily, since $U = L^{-1}Q$, as functions of distance along the boundary and time.

For subsonic inflow case, the first diagonal element of Λ is negative. This means that all except one characteristic can be specified. Thus for a given choice of state variables $U = U(s,t)$, where s is the distance along the boundary, a left eigenvector matrix L can be constructed. The specified characteristics are $q_2, q_3, \ldots, q_{N_e}$ as explained below. As indicated in Figure (7.8) the variable q_1^N is interpolated from the interior domain based upon L at the inlet. The characteristic direction implies that the position I from where information at the inlet node N be gathered is given by

$$-\Delta s \;=\; (V - a_f)\Delta t. \tag{7.35}$$

This position lies along a streamline and at a distance Δs from the boundary node B. Note that again if the CFL constraint is satisfied the position I would lie within one of the two cells adjacent to node B. Further note that the characteristic direction is not along the segment I-N (except for frozen flow) but that the variation of characteristic q_1 is known along this segment. Thus the characteristic variable corresponding to an updated node N is

$$q_1^N \;=\; q_1^I + z_1^I \Delta t \tag{7.36}$$

where interpolated values at location I are computed from the corner node values of the cell in which this location is determined, *i.e.*, with known values of U^I at previous time-level the following values can be calculated

$$q_1^I \;=\; \sum_{j=1}^{N_e} L_{1j}^B U_j^I\,, \qquad z_1^I \;=\; \sum_{j=1}^{N_e} L_{1j}^B W_j^I. \tag{7.37}$$

Note that the locally frozen (constant) values of the eigenvector are based upon the

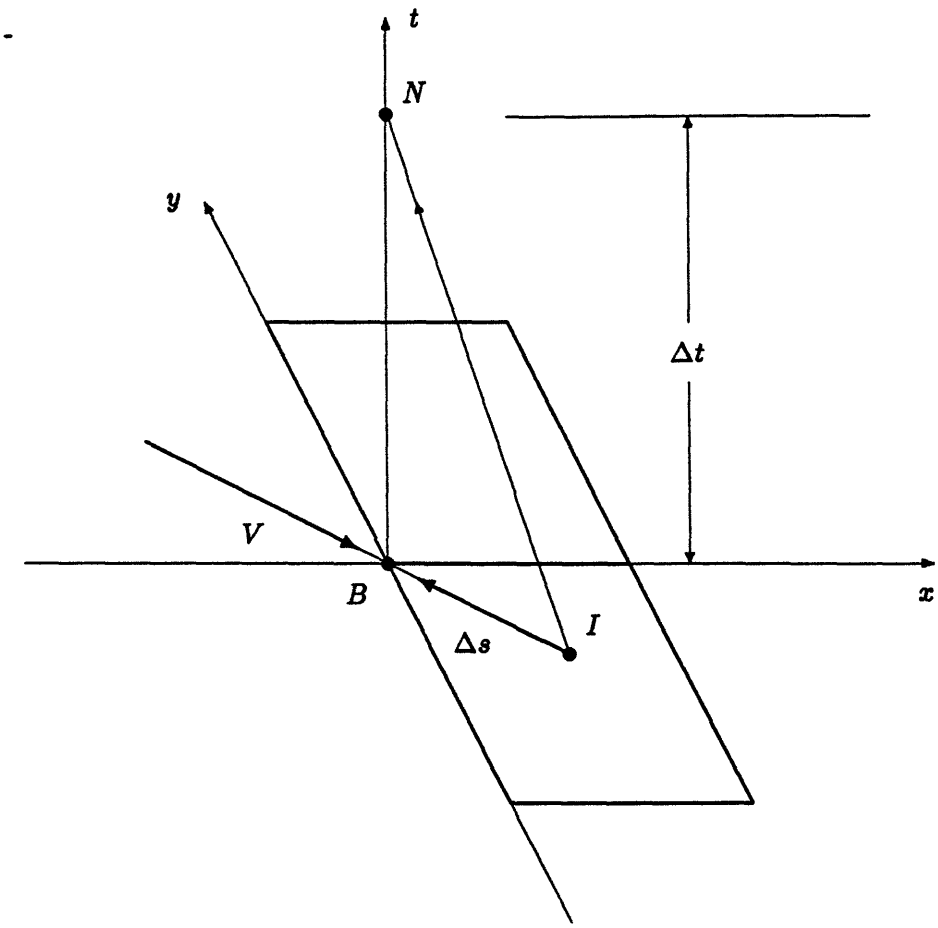

Figure 7.8: Characteristic subsonic inflow boundary condition.

values at node B. The other characteristic variables are given by

$$q_i = \sum_{j=1}^{N_e} L_{ij}^B U_j^B , \qquad i = 2, \ldots, N_e. \tag{7.38}$$

Now that the vector $Q^N = (q_1^N, q_2, \ldots, q_{N_e})$ is determined the state vector U^N can be calculated from the inverse relation

$$U^N = (L^{-1})^B Q^N. \tag{7.39}$$

7.5.3 Outflow Boundaries

For supersonic outflow all eigenvalues are positive which implies that all information must propagate from the interior to the exit plane and the conditions outside the exit plane have no influence on the interior flow. This means that, unlike the solid wall boundary condition, there is no need to consider "ghost" cells from the exterior domain; their contribution is zero and the change at the interface need not be multiplied by a factor of two. The characteristic for some node B (at time t and node N at time $t+\Delta t$) is

$$Q^N = L^B(U^B + \delta U^B) \tag{7.40}$$

where δU^B is the contribution at node B for all cells adjacent to it as predicted by the Ni scheme. Premultiplication by the inverse eigenvector matrix yields the correct change at node B as the value predicted by the Ni scheme. Hence no special treatment is needed for supersonic exit boundary.

For subsonic outflow the eigenvalue $(V - a_f)$ is negative and all others are positive. Hence only one physical parameter can be prescribed at this boundary. A typical choice for this parameter is the back pressure $p_b(s,t)$ that controls the flow at exit. Therefore a consistent physical condition can be formulated if the first characteristic is based upon the back pressure and the current state values (minus one) at the exit node and the rest of the characteristics from the interior. Thus the energy term component ϵ_b^B of the state vector (and temperature) can be recomputed by using the back pressure p_b and other known values at the exit node B. The first characteristic is computed as

$$q_1 = \sum_{\substack{j=1 \\ j \neq 4}}^{N_e} L_{1j}^B U_j^B + L_{14}\epsilon_b^B . \tag{7.41}$$

The other characteristics originate from the interior domain and, corresponding to updated node N, are given by

$$q_i^N = q_i^I + z_i^I \Delta t , \qquad i = 2, \ldots, N_e \tag{7.42}$$

where the location I is given by the distance

$$\Delta s = -(V + a_f)\Delta t \tag{7.43}$$

along the streamline passing through node B for $i = 2$ and by the distance

$$\Delta s = -V\Delta t \tag{7.44}$$

along that streamline for $i = 3, \ldots, N_e$. The expressions for q_i^I and z_i^I are given by Equations (7.37) with appropriately interpolated values for location I. Once the vector $Q^N = (q_1, q_2^N, \ldots, q_{N_e}^N)$ is determined the state vector U^N can be calculated from the inverse relation as given by Equation (7.38).

Chapter 8

Results

The results in this chapter are divided into three sections. The first section contains one-dimensional results for relaxing shock tubes and steady state streamtube flows. The blast waves for two-dimensional flows are included in the second section. The considered geometries were a circular arc on the lower wall of a cascade configuration and a 90 degree bend duct. The medium is either a perfect gas or a Lighthill dissociating gas and a single shock propagates along the channel. The third section pertains to scramjet inlets. Examples include flow of a perfect gas through a two-strut inlet and a premixed hydrogen combustion model for the same geometry. Another simpler geometry is considered in which the inflow mass flow rate is varied sinusoidally and its influence is examined on the flow variables.

8.1 One Spatial Dimension

Three examples have been considered to illustrate the unsteady adaptive technique. These are

1. a converging-diverging streamtube with a single dissociating gas,
2. a shock-tube with a single dissociating gas,
3. a diverging channel with multiple reactions.

For all the examples the artificial viscosity coefficient is restricted to the interval $\sigma \in [0.01, 0.2]$ and the reacting flow cases have been carried out by the source implicit

($q = 0$) scheme. The CFL number for all steady state examples is 0.9 whereas that for the shock-tube cases is 0.7. The constants used to define the temporal resolution are $\epsilon_0 = 0.01$ and $\epsilon_1 = 0.05$ in the shock tube cases. Except for the diverging channel case all calculations were performed in single precision. For steady state applications the convergence criterion was based upon the root mean square (rms) error of the momentum term (except for the last case) and convergence was assumed when this error became less than 10^{-5}.

8.1.1 Converging-Diverging Streamtube

Consider first a Lighthill ideal dissociating gas, $Z_2 \rightleftharpoons 2Z$, which is assumed to be flowing through a converging-diverging streamtube with an area distribution of the form

$$A = 1 + 0.5x^2 \tag{8.1}$$

here x is a non-dimensional measure of distance from the throat in units of the throat height and the area A is normalized by the throat area.

An initial verification of the code consisted of shock free flow examples and comparison of the results with Bray [20] for several values of the reaction parameter, Φ, with a wide range of values between zero for frozen flow, to infinity for equilibrium flow. The results are for $x \in [-2, 5]$ and the dimensionless temperature and pressure of

$$\frac{T_i}{\theta_d} = 0.1 \ , \qquad \frac{p_i \hat{m}_Z}{\mathcal{R}\rho_d\theta_d} = 2.5 \times 10^{-6}. \tag{8.2}$$

The subscript i indicates the inlet which is very nearly the reservoir condition. The inlet values for temperature and pressure correspond to 5950 K, 115 atm for oxygen and 11300 K, 215 atm for nitrogen. The accompanying degree of dissociation Y_i and dimensionless density for equilibrium at inlet are

$$Y_i = 0.69 \ , \qquad \frac{\rho_i}{\rho_d} = 2.9561 \times 10^{-5}. \tag{8.3}$$

Compared to the definition of reaction parameter, Φ, as utilized here, Bray defined his reaction parameter, Φ_b, in a slightly different manner. The conversion between the two

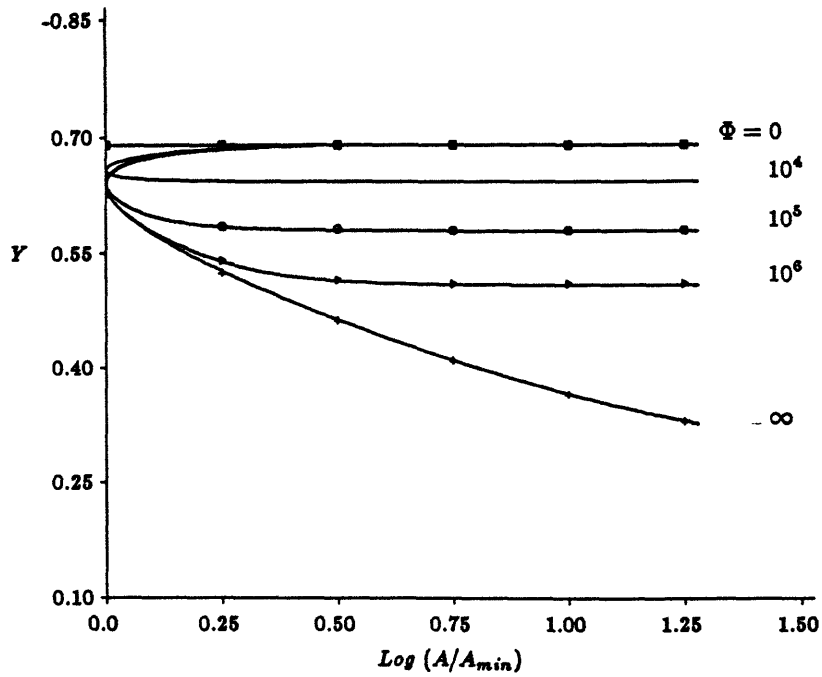

Figure 8.1: Degree of dissociation versus area ratio for several values of rate parameter Φ, symbols represent Bray's calculations, Reference [20].

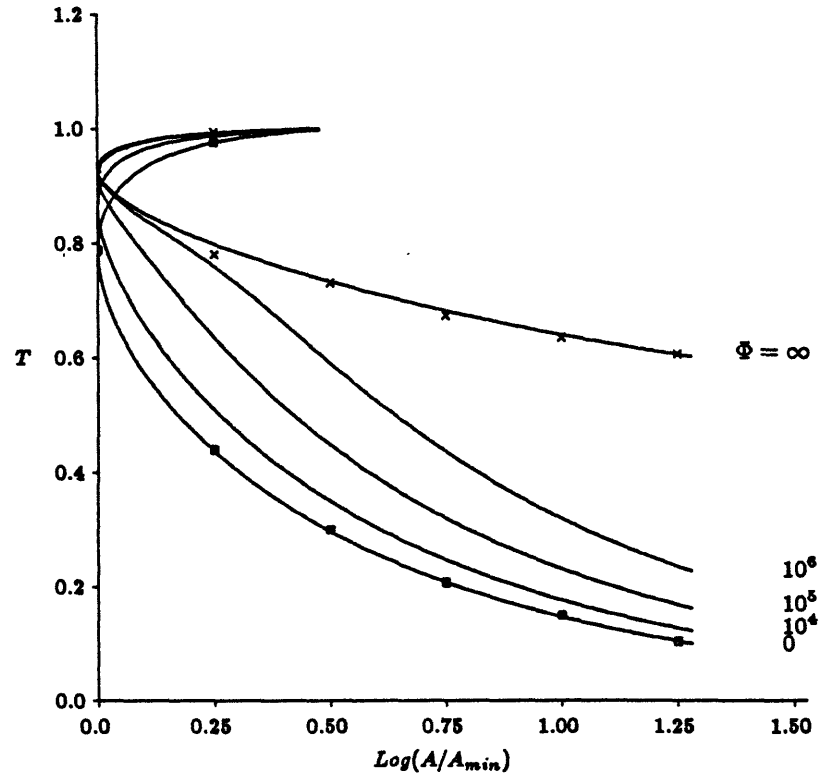

Figure 8.2: Temperature versus area ratio for several values of rate parameter Φ, symbols represent Bray's calculations, Reference [20].

reaction parameters is as follows:

$$\frac{\Phi_b}{\Phi} = \frac{\rho_d}{\rho_i}\sqrt{\frac{T_i}{\theta_d}\frac{(1+Y_i)}{8}} = 4917. \tag{8.4}$$

Figures (8.1) and (8.2) show steady-state results obtained with *local time-stepping* with a CFL number of 0.9 and a uniform grid. Specifically the degree of dissociation and temperature distributions appear on a plot folded about the minimum area section such that the upper curves correspond to the subsonic upstream region. The symbols in these Figures indicate Bray's calculations whereas the solid curves are the result of the present scheme. The criterion for temporal resolution, Equation (6.5), was not used in this case. Except for the frozen case, all curves fall rapidly in the vicinity of the throat. In the equilibrium solution ($\Phi \to \infty$), the mass fraction continues to drop in the supersonic flow and vanishes as the area ratio approaches infinity. It is also observed from Figure (8.1) that the solutions with finite dissociation rates are initially indistinguishable from the equilibrium curve, in the upstream part of the nozzle. The deviations begin near the minimum section and once these deviations from the local equilibrium conditions become appreciable the degree of dissociation approaches a constant value. In the corresponding equilibrium case the temperature continues to fall due to the divergence of the streamtube which triggers recombination of atoms into molecules. However, recombination becomes essentially *frozen* in the supersonic regions for intermediate Φ values. The temperature profiles indicate that freezing causes a very large reduction in temperature compared to the equilibrium solution. This is because the chemical energy associated with dissociation is not available for intermediate Φ values due to the higher degree of dissociation. The departure from equilibrium also reduces the flow velocity and for propulsive nozzles the freezing phenomenon results in a loss of thrust.

Figure (8.3) shows another steady flow through the same parabolic nozzle but for a curtailed domain $x \in [-2, 2]$. The reaction parameter is $\Phi = 10^4$ and a back pressure ratio $p_b/p_i = 0.92$ is specified, so that a normal shock would be stationed at $x = 0.5$ for a frozen flow situation. Two levels of spatial embedding and local time-stepping were used for the adapted case. Temporal resolution was only based upon the CFL restric-

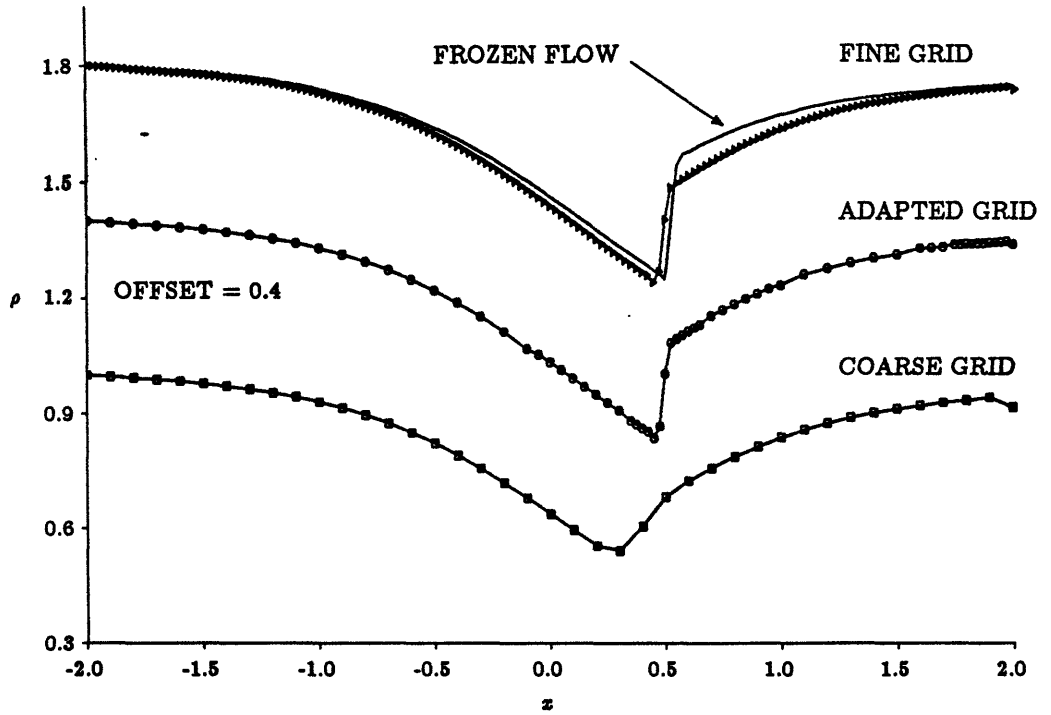

Figure 8.3: Density variation of flow through a converging-diverging streamtube with $\Phi = 10^4$ for coarse, adapted and fine grids.

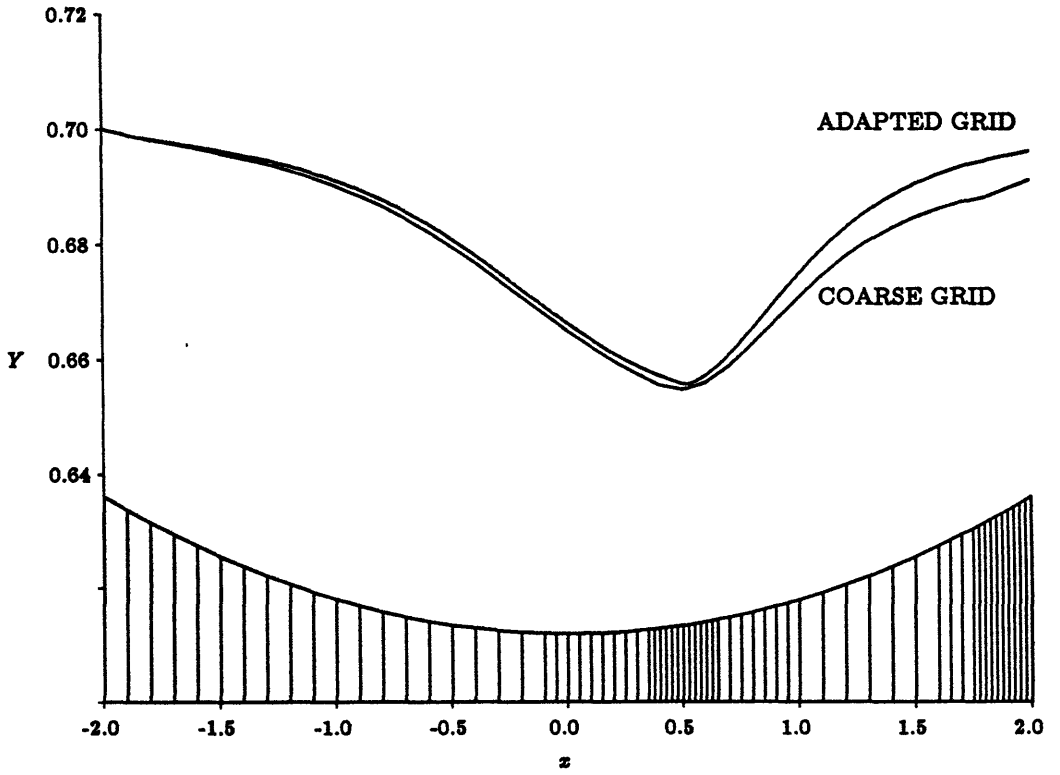

Figure 8.4: Degree of dissociation versus x-location for converging-diverging streamtube with $\Phi = 10^4$ for coarse and adapted grid and the spatial grid variation.

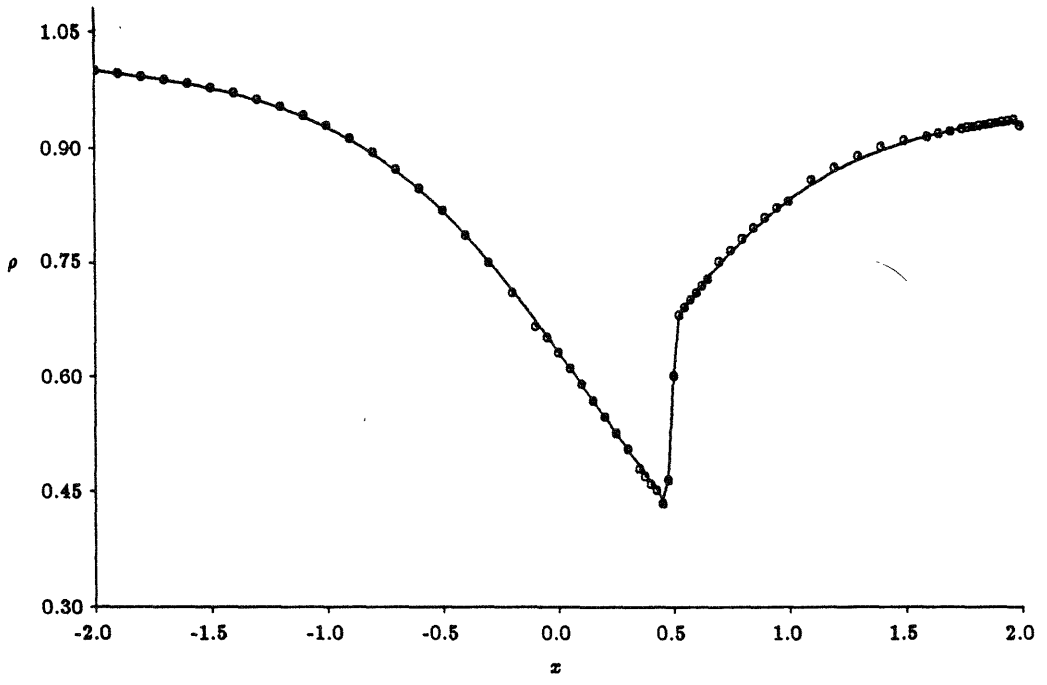

Figure 8.5: Density variation through a converging-diverging streamtube for fine and adapted grids, $\Phi = 10^4$.

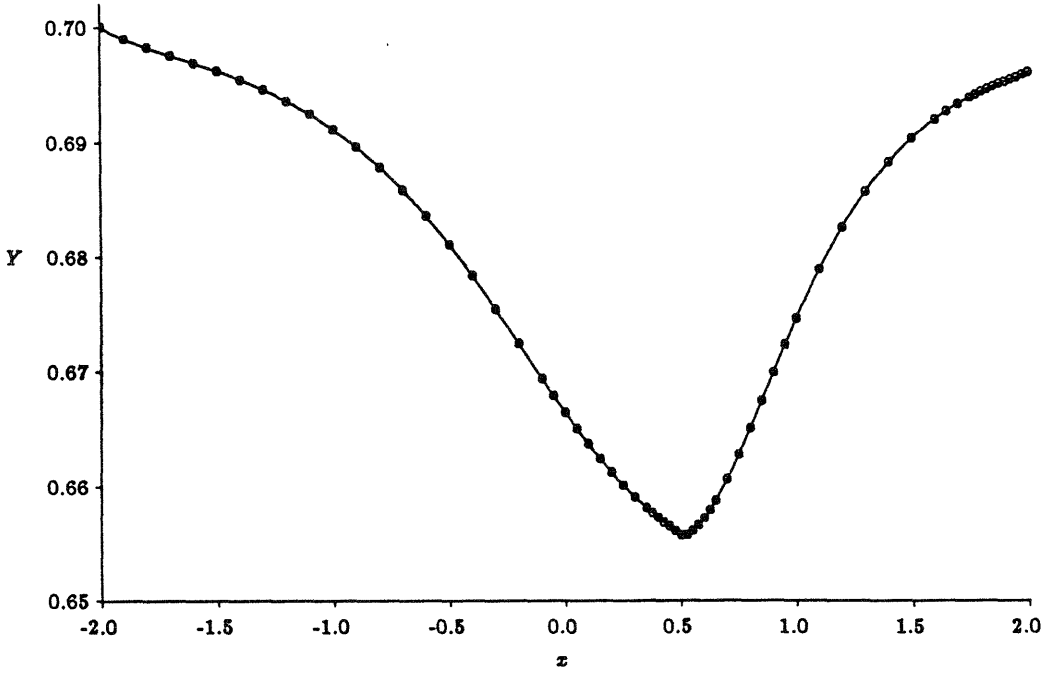

Figure 8.6: Variation of degree of dissociation through a converging-diverging streamtube for fine and adapted grids, $\Phi = 10^4$.

tion. Spatial resolution was based upon first differences of density with the first divide threshold value of $R_{d1} = 1.2$ and the second threshold value R_{d2} was calculated to be the limit for which atmost 20% ($C_{fd} = 0.2$) of the cells would be divided, (see Section 5.3.3 for more details). The results are shown corresponding to coarse, embedded and fine grids, with relative computing times 10.1 (fine/coarse) and 1.4 (adapted/coarse). The vertical scale corresponds to the coarse grid and the other two curves are displaced by the indicated offset. Each symbol in the figure corresponds to a computational node in the domain and the placement of these symbols indicates the type of grid utilized. For comparative purposes the density distribution for the corresponding frozen flow is shown as a curve without symbols along with the fine grid relaxing solution. Shown in Figure (8.4) is the final grid and degree of dissociation for both adapted and coarse grid cases, and indicates that the coarse grid solution predicts an appreciably different degree of dissociation aft of the normal shock.

The embedded and fine grid solutions agree very well, as is evident in Figures (8.5) and (8.6), whereas the shock location is displaced and spread out for the coarse grid. These figures also indicate that the normal shock occurs before the freezing phenomenon has been completed; however the flow also is not in equilibrium ahead of the shock, as can be read from Figure (8.1). The normal shock increases the temperature and decreases the velocity which allows further dissociation. Unlike the freezing phenomenon in the supersonic region, the flow after the shock gradually approaches the corresponding equilibrium state as can be seen from the relaxation behind the shock. Due to the divergence of area the velocity continues to decrease and the final equilibrium state becomes very nearly equal to that at the inlet. In principle, the two equilibrium values need not be the same due to the stagnation pressure loss across the normal shock. However, in this particular case the shock is very close to the minimum section and the stagnation pressure loss is small. Hence the corresponding equilibrium degree of dissociation at the exit is slightly less compared to the value at inlet and the flow in the trailing part of the nozzle is essentially in equilibrium.

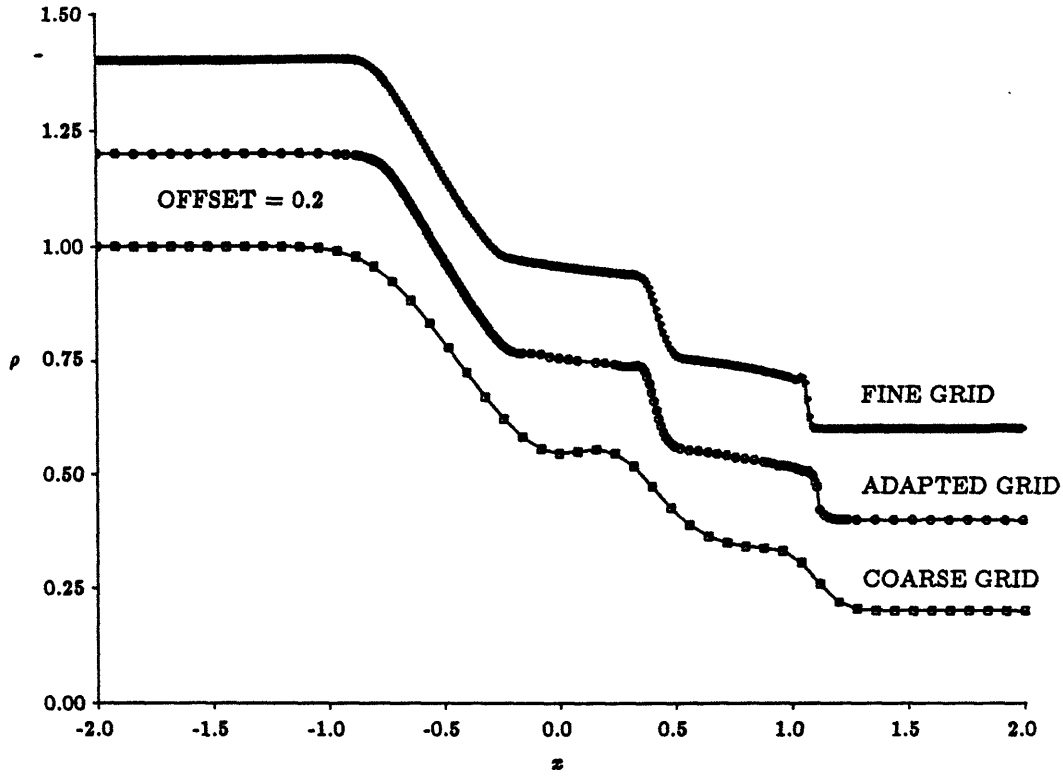

Figure 8.7: Non-equilibrium shock tube flow for $\Phi = 10^4$ on coarse, adapted and fine grids at $t = 0.6$, symbols show computational nodes.

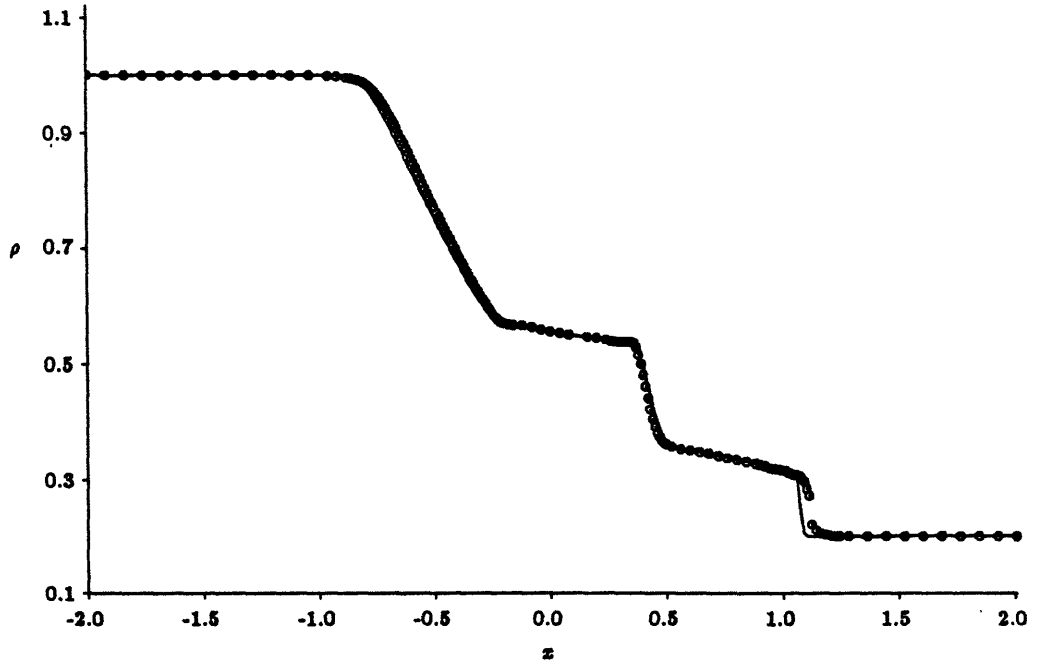

Figure 8.8: Overlay of fine and adapted grid solutions at $t = 0.6$ for $\Phi = 10^4$, solid curve is fine solution and symbols indicate computational nodes for adapted case.

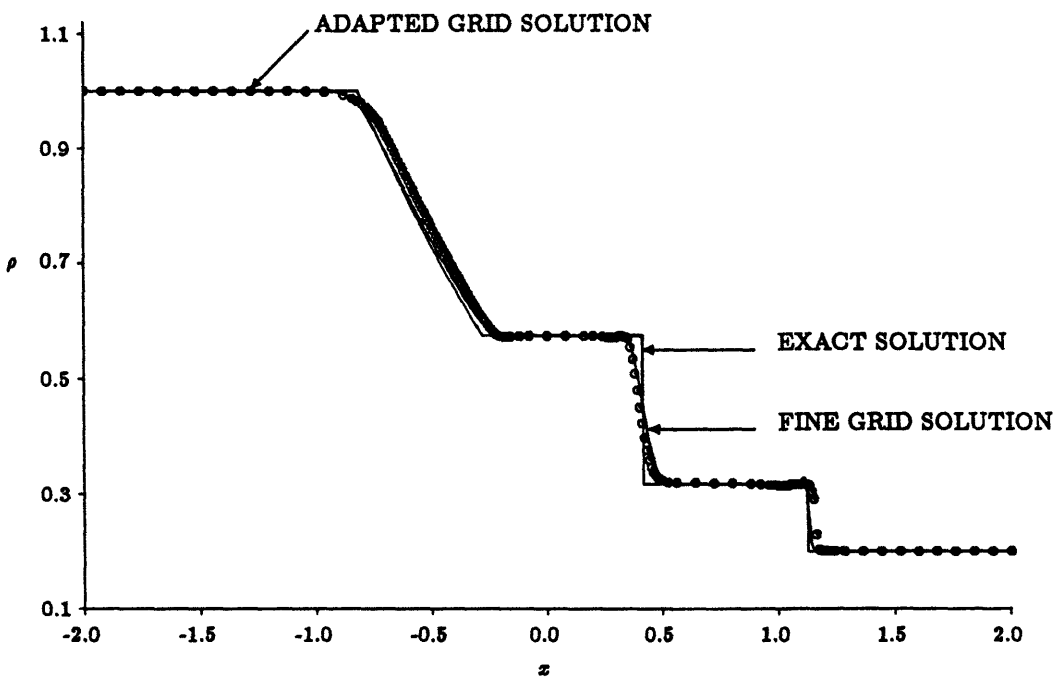

Figure 8.9: Frozen shock tube solution for adapted and fine grids alongwith the exact solution at $t = 0.6$.

	$\Phi = 0$	10^4	10^5
Coarse	1.00	1.21	1.89
Adapted	6.54	8.77	14.40
Fine	49.95	65.38	110.81

Table 8.1: Comparison of CPU time for shock tube calculations.

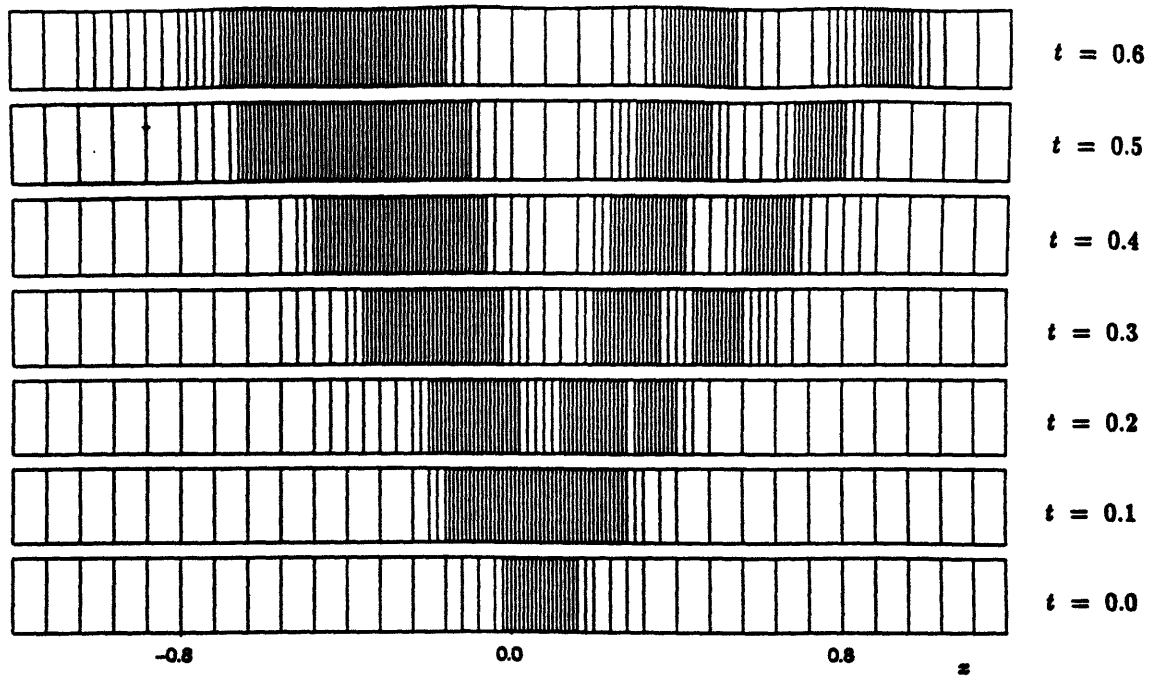

Figure 8.10: Evolution of spatial grid for frozen shock tube flow.

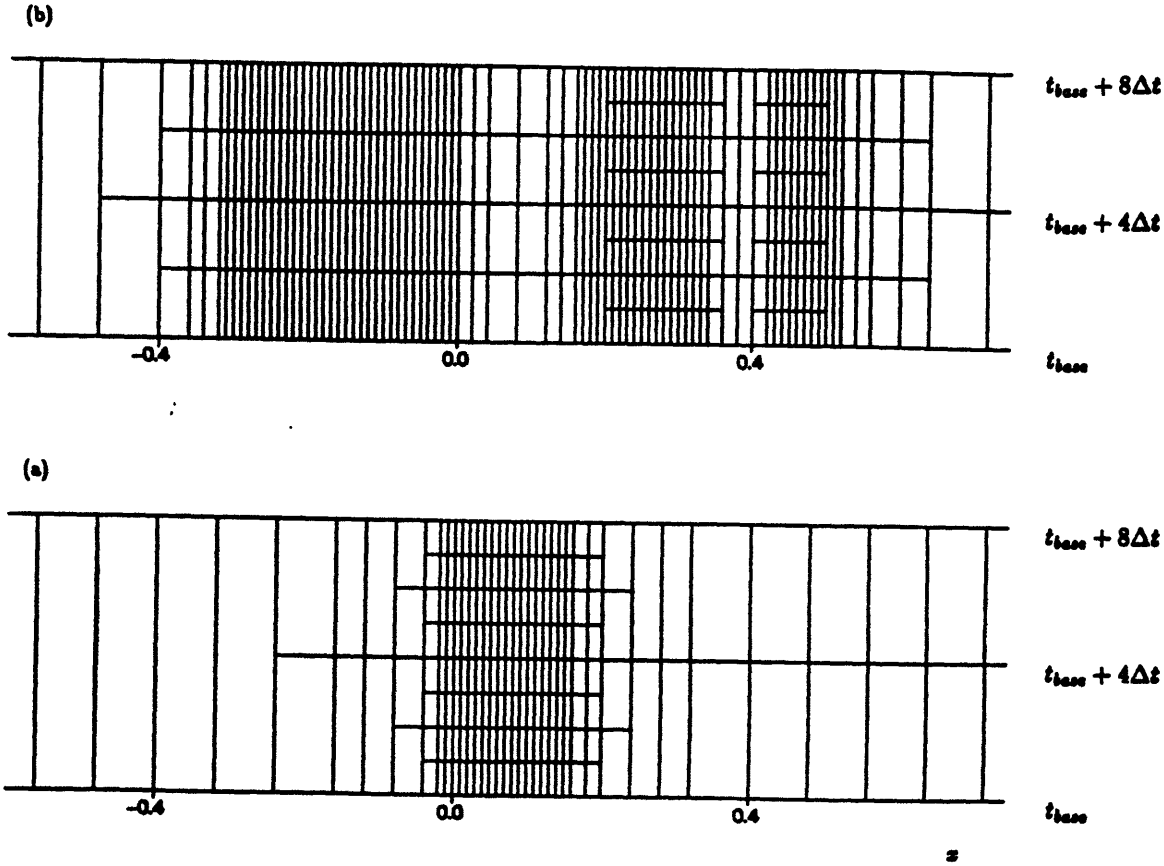

Figure 8.11: Evolving temporal grid for frozen shock tube near $t = 0$ and $t = 0.2$: (a) $t_{base} = 0$, $\Delta t = 2.9075 \times 10^{-3}$, (b) $t_{base} = 0.2079$, $\Delta t = 2.8787 \times 10^{-3}$.

8.1.2 Constant Area Shock Tube

A second example was carried out for unsteady shock tube flow for both frozen and reacting cases, ($\Phi = 0, 10^4$). The initial conditions across the contact surface were $p_e/p_i = 0.2$, $T_e/T_i = 1.0$ where stations i, e indicate inlet, exit of the computational domain which are respectively the high, low pressure sides. For the frozen case the temporal resolution was only based upon the CFL constraint ($\Gamma = 0.7$) whereas this constraint and resolution based upon mass fraction of dissociated atoms was used for the reacting case. Three levels of both spatial and temporal adaptations were introduced and the final results shown correspond to $t = 0.6$. Figure (8.7) indicates the density variations for $\Phi = 10^4$ at the final time for coarse, adapted and fine grids and the curves are offseted for clarity. The symbols on these figures indicate the computational nodes in the domain. We again note that the coarse grid solutions are poorer than either the fine or adapted grid solutions. The non-uniform distribution between the frontal shock and the contact surface as well as that between the contact surface and the trailing edge of the expansion fan indicate relaxing regions within which dissociation is taking place. The overlay of the adapted and fine grid solutions at $t = 0.6$ for $\Phi = 10^4$ is shown in Figure (8.8). The comparison is reasonably good except that the shock speed for the adapted case is overpredicted by about 3%. A similar overlay at $t = 0.6$ for the frozen case is shown in Figure (8.9) where the adapted grid overpredicts the shock speed by about 2% compared to the exact solution. Since the shock speed predicted by the two-dimensional spatio-temporal algorithm does not exhibit this behavior, it is conjectured that the error in shock speed in the current one-dimensional algorithm is due to the non-uniformity parameter ϵ_j pre-multiplying the flux change ΔF_{jC} in Equation (3.35). Note that, although the inclusion of these terms yields higher order accuracy, they may in fact adversely affect the solution near strong shocks due to their non-conservative nature.

Figure (8.10) shows the progression of grids as time increases for the frozen case, whereas Figure (8.11) shows the evolving temporal grid near time levels, $t = 0$ and $t = 0.2$ for this case. The spatial grid clearly tracks the expansion fan, contact surface and the shock wave. The time-grid shows eight smallest time-steps in each time-stride

and that the separation between consecutive isotemporal surfaces in fact is not constant. The temporal grid also indicates cell locations with finer temporal resolution which correspond to relatively coarser spatial resolution and *vice versa*. Although the concept is straight-forward, the time-grids become very complicated for two spatial dimensions and will not be shown henceforth.

Figure (8.12) indicates the evolution of density on the adapted grid for the frozen flow and the exact solution. The evolution of density and atom mass fraction for the dissociating case is shown in Figures (8.13) and (8.14). Although the results corresponding to $\Phi = 10^5$ are not shown here, the CPU time comparisons for $\Phi = 0, 10^4, 10^5$ are indicated in Table (8.1) to show the effectiveness of the procedure compared to the global approach. The advantage of the current spatio-temporal algorithm is clearly seen to increase as the stiffness level increases.

For the dissociating gas the initial ($t = 0$) degree of dissociation is regarded as constant. The corresponding equilibrium degree of dissociation for $x \geq 0$ at $t = 0$ is $Y = 0.90$. As the contact surface is allowed to move the flow ahead of the shock stays quiescent and hence the dissociation level is not changed. However, the flow just after the shock finds itself to be deviated from the corresponding equilibrium conditions and starts relaxing behind it. As the residence time behind the shock increases, the relaxation region grows in between the shock and the contact surface and the degree of dissociation is seen to increase gradually behind the shock. At $t = 0.6$ the maximum degree of dissociation is observed to be about $Y = 0.8$; there is no reason for the atom mass fraction to reach the value $Y = 0.9$, since the conditions just after the shock for $t > 0$ have changed. The dissociation level through the expansion fan gradually decreases from the leading edge to the trailing edge and continues decreasing at the same rate in between the trailing edge and contact surface. Hence the trailing edge of the expansion fan cannot be easily identified by examining the degree of dissociation plots. The largest change in the degree of dissociation is experienced across the contact surface.

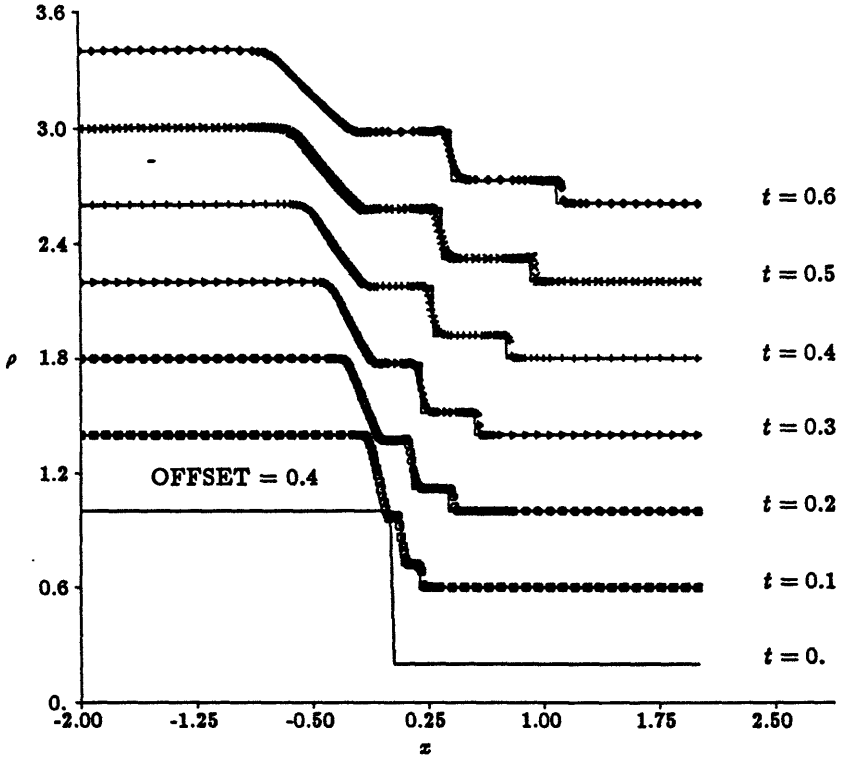

Figure 8.12: Evolution of density for frozen shock tube flow on adapted grids, solid curves indicate exact solution.

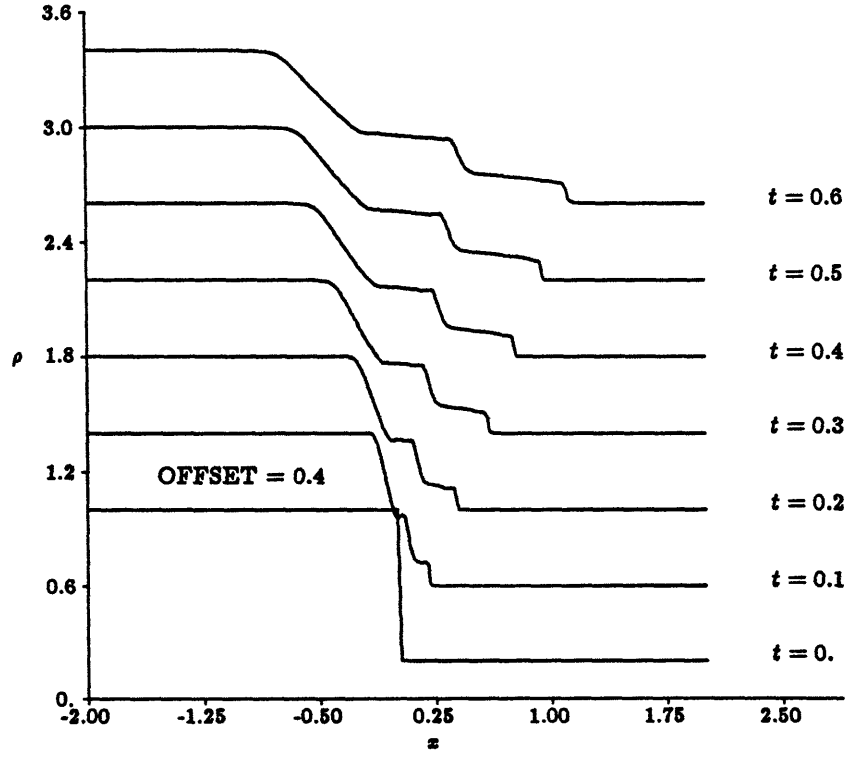

Figure 8.13: Evolution of density for reacting shock tube flow on adapted grids, $\Phi = 10^4$.

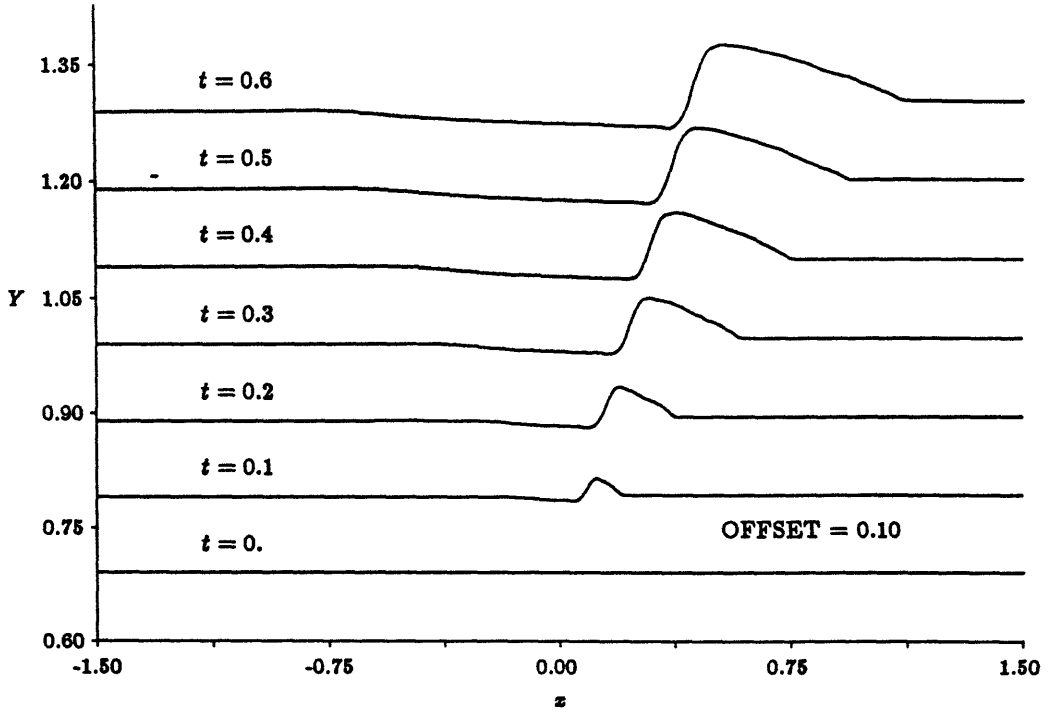

Figure 8.14: Evolution of degree of dissociation for shock tube flow on adapted grids, $\Phi = 10^4$.

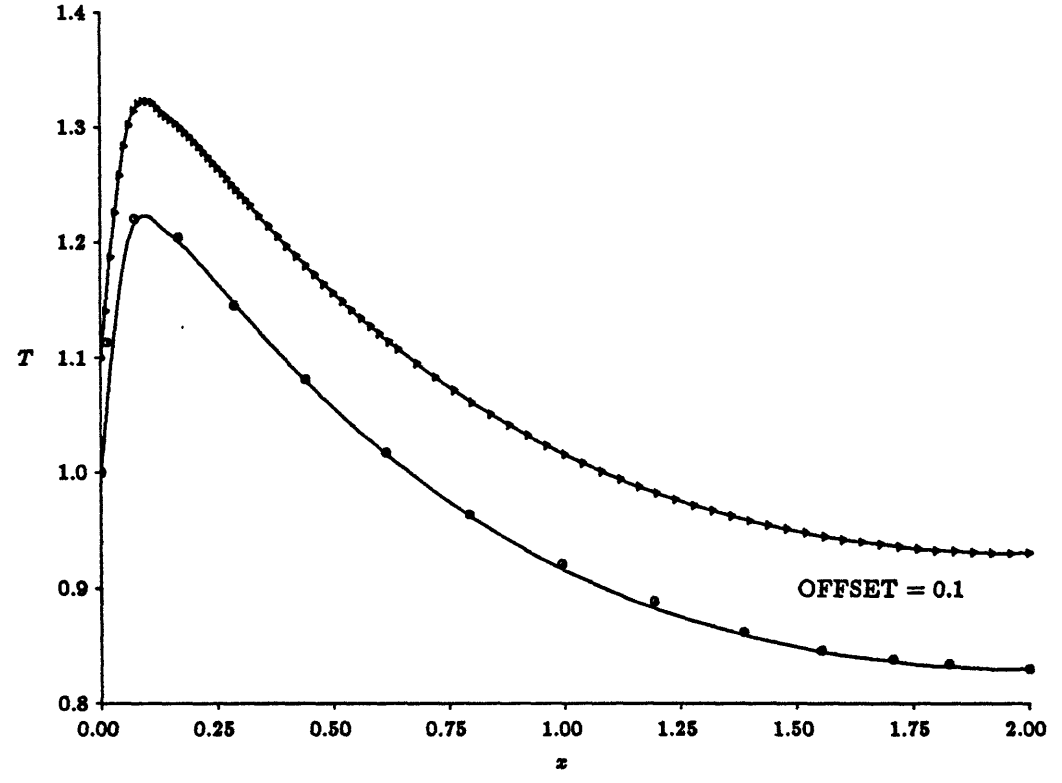

Figure 8.15: Temperature profile for diverging channel, both solid lines indicate current calculations and triangles indicate computational nodes in the current scheme, circles represent computations from Reference [42].

8.1.3 Diverging Streamtube

As a final example for flow in one spatial dimension, the Rogers and Chinitz model (Section 2.7) for the streamtube with area distribution

$$A = \left[1 + \sin(\frac{\pi x}{4})\right]^2 \tag{8.5}$$

was considered. The area is again nornmalized by the throat (inlet) area. The reference or inlet conditions were

$$T_r = 1900K, \quad p_r = 81000Pa, \quad M = 1.4, \quad L_r = 1m, \quad \phi = 0.3 \tag{8.6}$$

where ϕ is the equivalence ratio. A schematic of the rapid expansion diffuser is shown in Figure (8.15). The same case was calculated by Drummond, Rogers and Hussaini [42]. The inlet conditions to this streamtube imply high concentration gradients near the inflow boundary and hence provide a formidable test for the algorithm. The total number of global nodes was chosen to be 51 with two levels of spatial and ten levels of temporal embedding and the calculations were carried out to steady state. The convergence criterion was based upon an rms error of mass fraction of hydrogen and the calculations were continued until the error was reduced by eight orders of magnitude. Temporal resolution was based upon limiting the changes in the mass fraction of hydroxyl, according to Equation (6.5). This species was chosen because it is involved in both the reactions and its production rate due to the first reaction can be very high. The calculations took 4259 seconds on a MicroVAX-II. Reference [42] used a source implicit scheme with 101 grid points and the calculations took 2524 seconds to converge on a CYBER-175. Note that the comparative fine grid solution of the current case had 201 grid points and was estimated to take three orders of magnitude longer on the MicroVAX-II. Assuming a conservative estimate of 20 for the speed ratio between the CYBER and MicroVAX, the present results are obtained about 50 times faster for the same spatial grid resolution when compared to that of Reference [42]. Figure (8.15) shows the temperature distribution as solid lines for the current algorithm. The circles indicate the calculations of Drummond *et. al.* and the triangles indicate the computational grid utilized by the current approach. Other final results in terms of distributions of mass fractions of hydrogen, hydroxyl and steam (H_2O) are shown in Figures (8.16) and (8.17). The

symbols in these figures indicate the calculations of Reference [42] whereas the solid curves indicate the present calculations. These results are in fair agreement with those of Reference [42] and the differences are less than 3%.

8.2 Blast Waves in Two Spatial Dimensions

Numerical experiments were carried out for two channel geometries: (1) a circular arc convex surface on the lower surface of a cascade configuration, and (2) a 90 degree bend duct. In each case a single shock propagates along the channel. The medium was either a perfect gas or a non-equilibrium Lighthill gas.

For all the examples the artificial viscosity coefficient is restricted to the interval $\sigma \in [0.05, 0.5]$ and the dissociating flow cases have been carried out by the source implicit ($q = 0$) scheme. The CFL number for all examples is 0.7. The constants used to define the temporal resolution are $\epsilon_0 = 0.01$ and $\epsilon_1 = 0.05$.

8.2.1 Frozen Bump Case

The first example is for a frozen medium ($\gamma = 1.4$) and a shock moving at $M_f = 2$ past a 15% circular arc bump as shown in Figures (8.18-8.24) corresponding to the time periods of $t = 0, 0.2, 0.4, 0.8, 1.0, 1.2$ respectively. The channel dimensions are normalized by the chord length and is spanned by $x \in [-1, 2]$, $y \in [0, 0.8]$ as shown in Figure (8.24). The shock is initially ($t = 0$) at $x = -0.5$. The figures show both density contours and the corresponding spatial grids at the indicated time intervals. It is clear that the evolving spatial grid tracks the salient features. Three levels of spatial embedding beyond the base grid, and four levels of temporal strides were used. Note that the maximum eigenvalue $(u + a)$ varies significantly across a moving shock, *i.e.*,

$$\frac{(u+a)_e}{(u+a)_i} = \frac{\sqrt{\rho_i p_e / \rho_e p_i}}{M_i + 1}. \tag{8.7}$$

This value for $M_s = 2$ is 0.3923 and hence it is appropriate to use one additional level for temporal adaptation compared to that for the spatial adaptation for this frozen flow

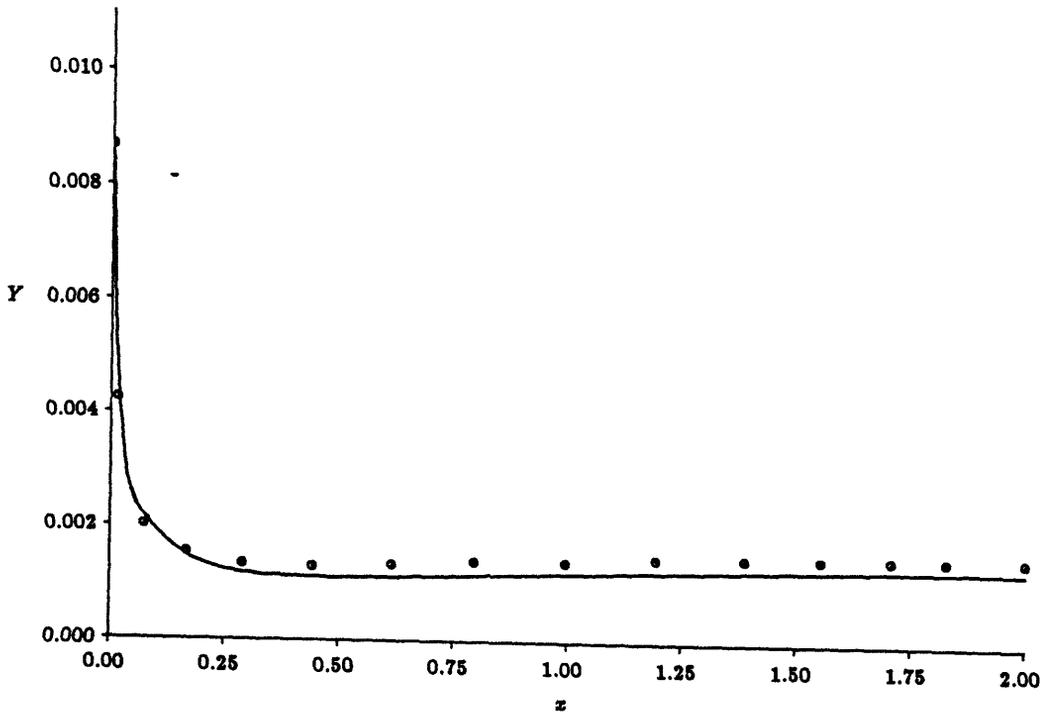

Figure 8.16: Hydrogen mass fraction profile for diverging channel, symbols represent computations from Reference [42].

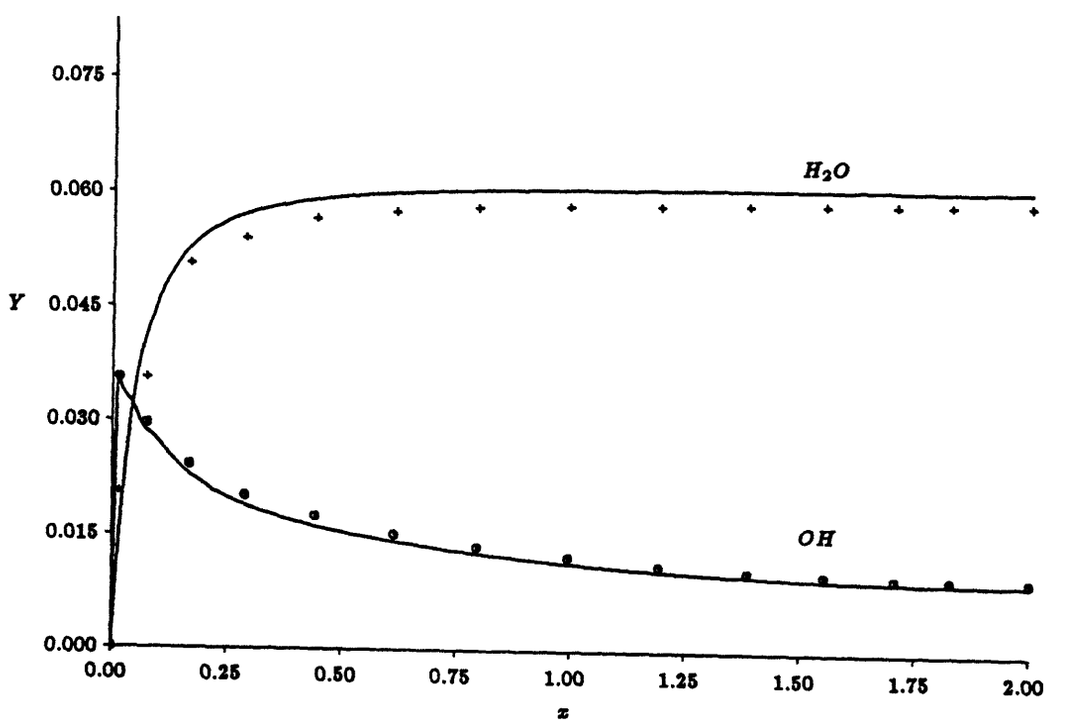

Figure 8.17: Hydroxyl and steam mass fraction profiles for diverging channel, symbols represent computations from Reference [42].

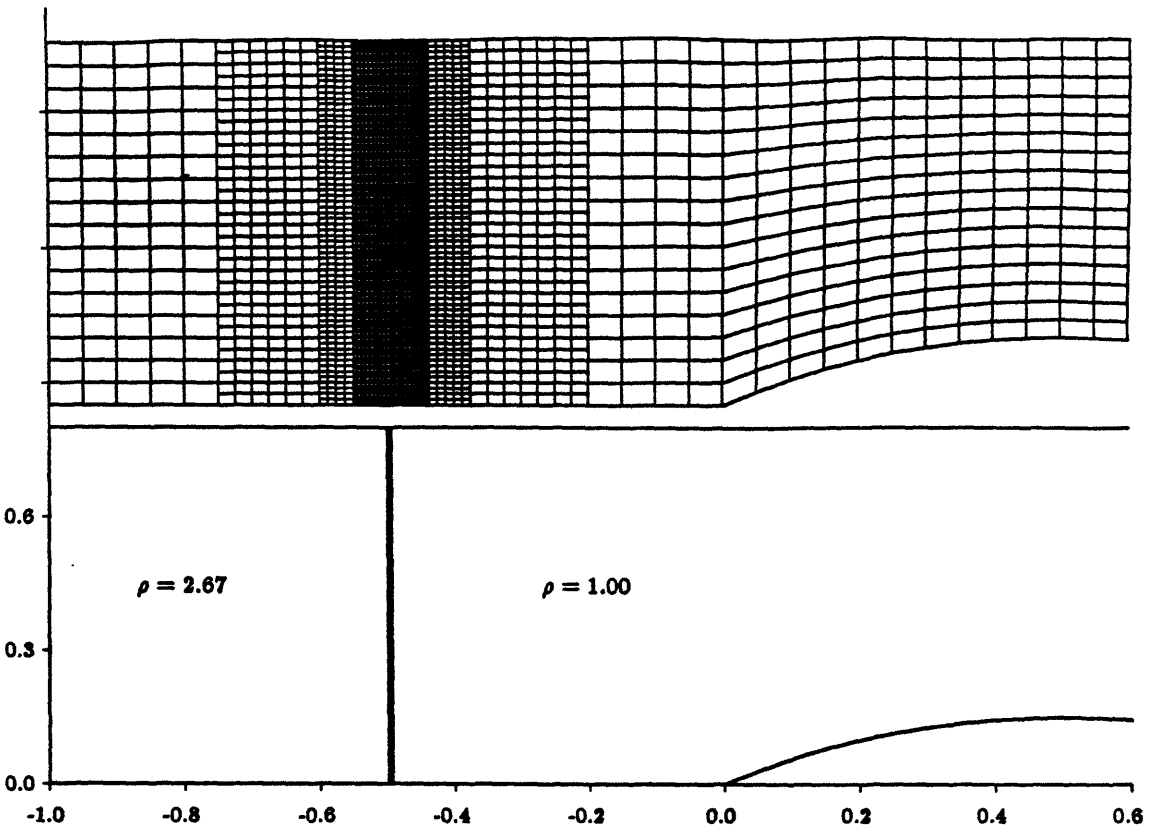

Figure 8.18: Grid and density contours at $t = 0$ for frozen flow over 15 % circular arc bump, $M_f = 2$.

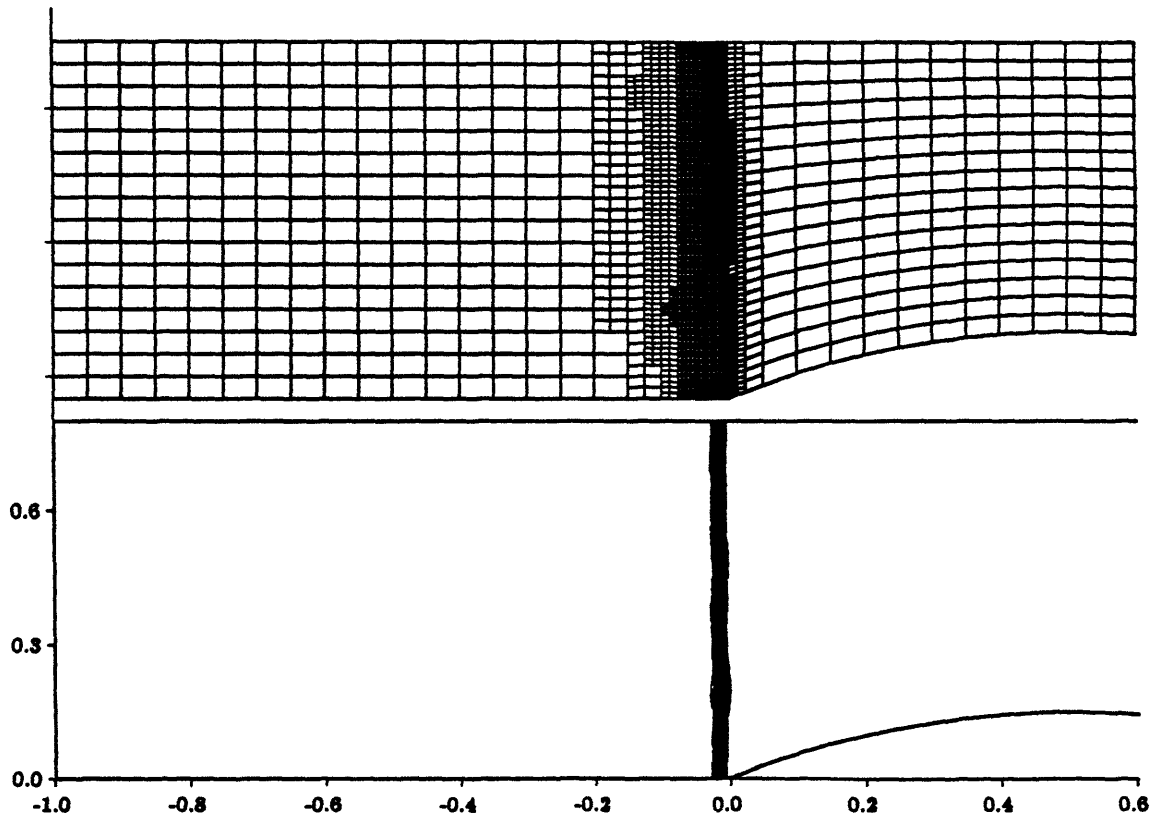

Figure 8.19: Grid and density contours at $t = 0.2$ for frozen flow over 15 % circular arc bump, $M_f = 2$.

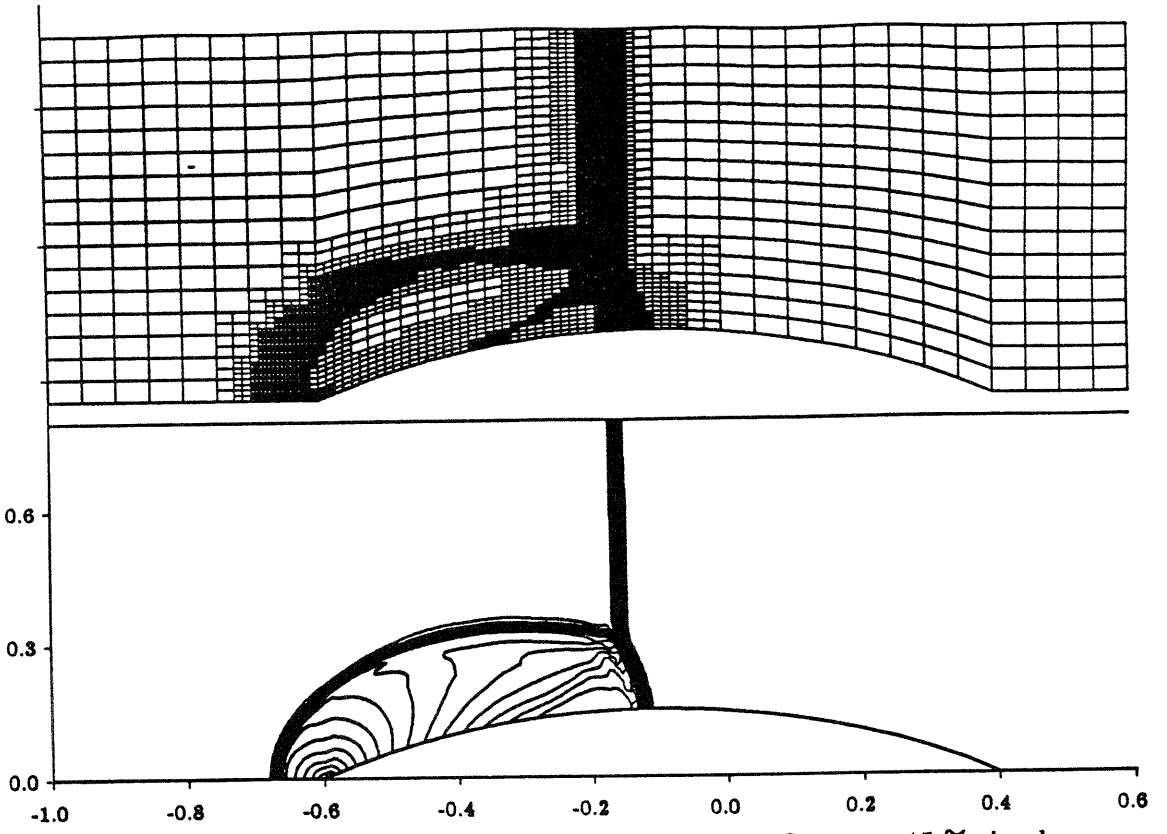

Figure 8.20: Grid and density contours at $t = 0.4$ for frozen flow over 15 % circular arc bump, $M_f = 2$.

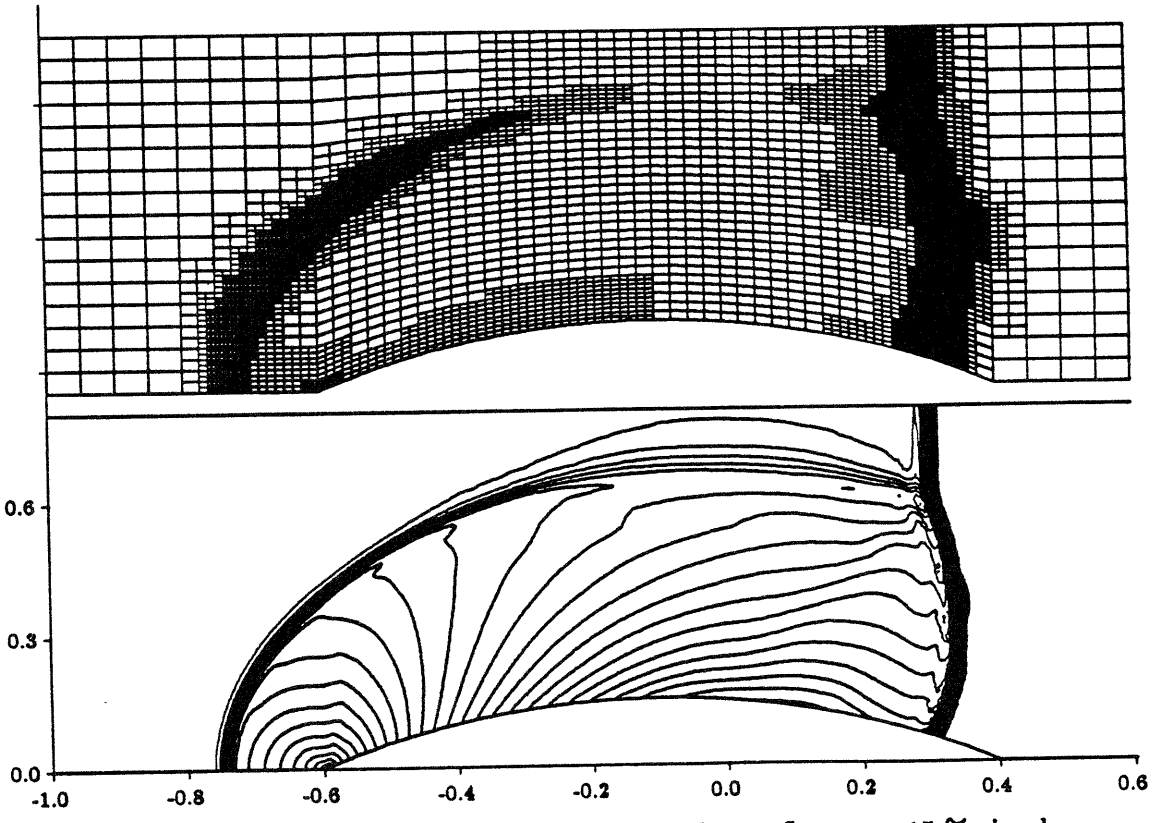

Figure 8.21: Grid and density contours at $t = 0.6$ for frozen flow over 15 % circular arc bump, $M_f = 2$.

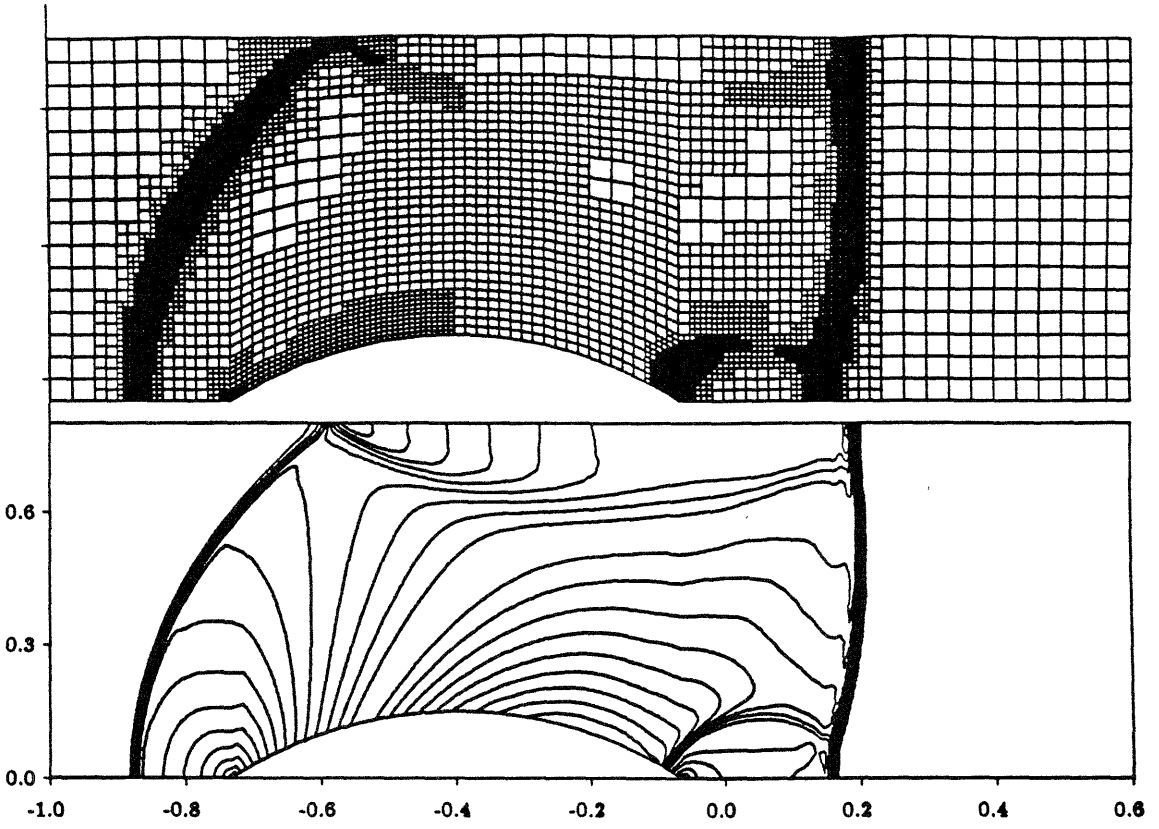

Figure 8.22: Grid and density contours at $t = 0.8$ for frozen flow over 15 % circular arc bump, $M_f = 2$.

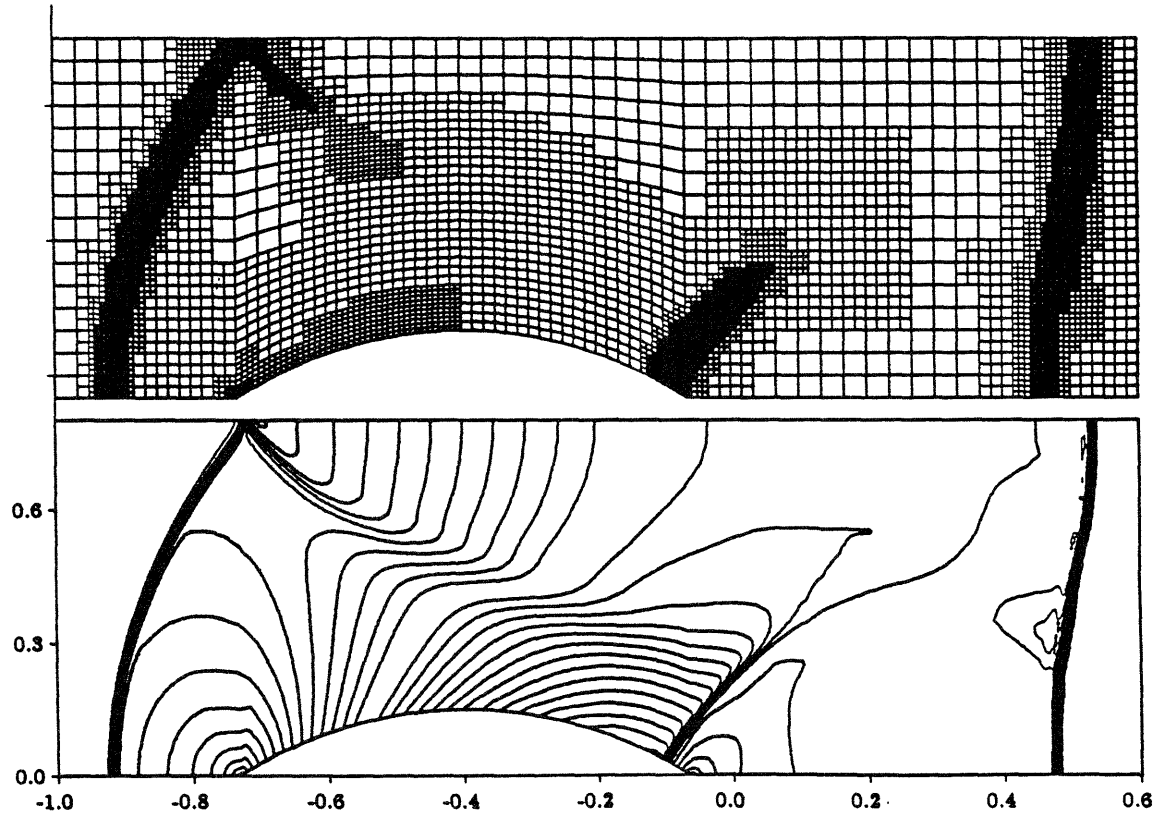

Figure 8.23: Grid and density contours at $t = 1.0$ for frozen flow over 15 % circular arc bump, $M_f = 2$.

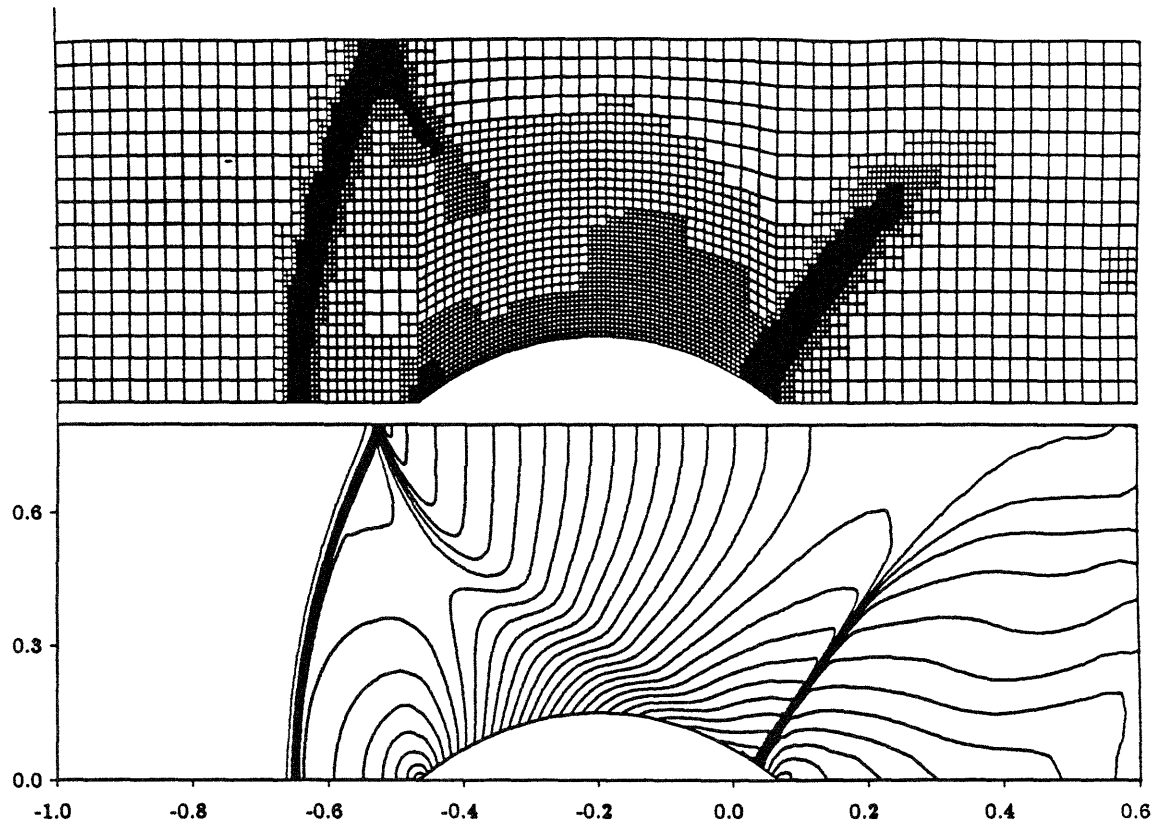

Figure 8.24: Grid and density contours at $t = 1.2$ for frozen flow over 15 % circular arc bump, $M_f = 2$.

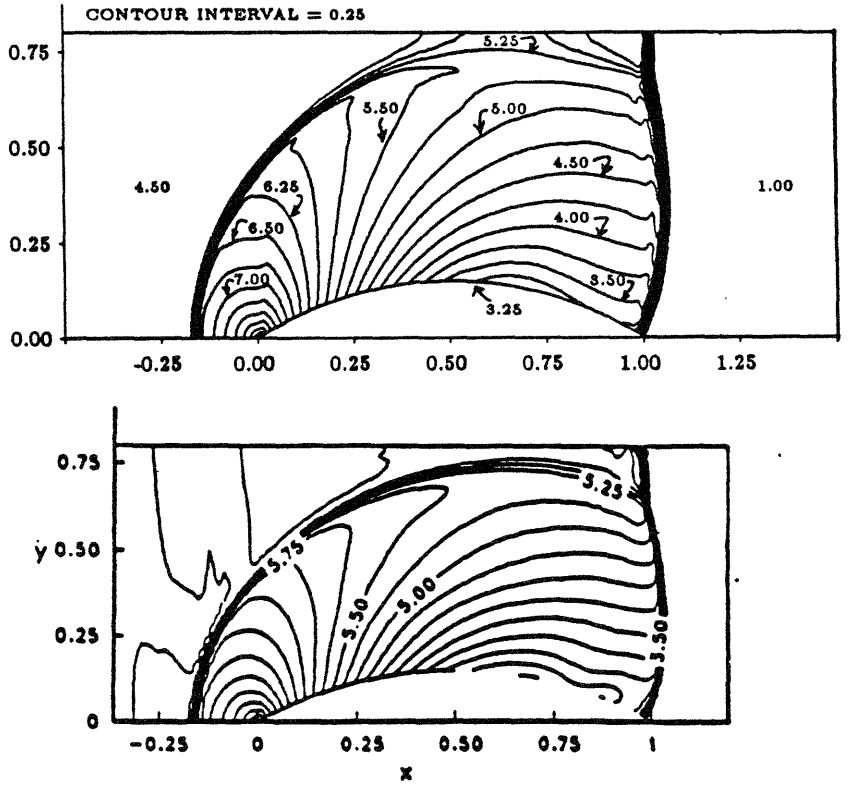

Figure 8.25: Pressure contours at $t = 0.65$ for frozen flow over 15 % circular arc bump, $M_s = 2$, (a) current calculation, (b) Yang *et.al.* [144].

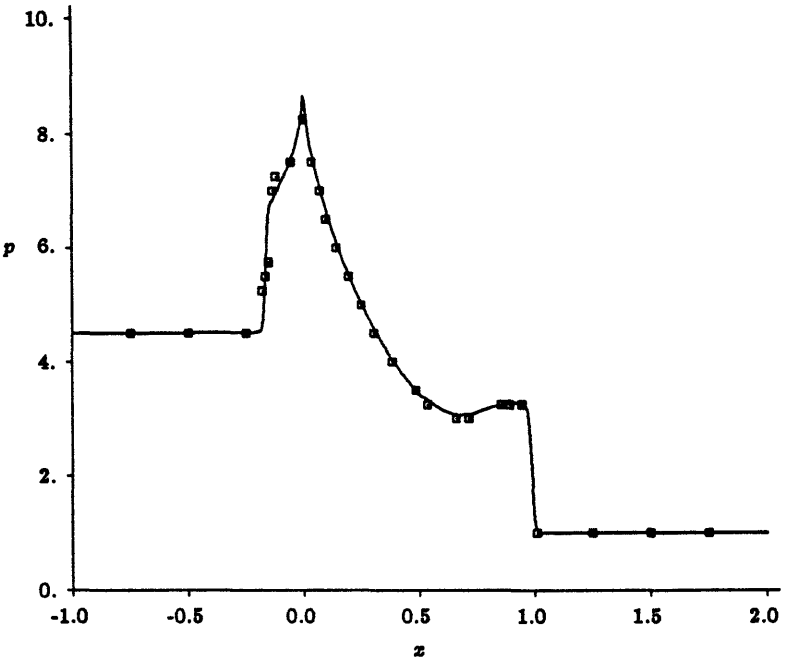

Figure 8.26: Comparison of pressure distribution on lower channel wall for frozen flow over 15 % circular arc, symbols represent Yang's calculations, Reference [144].

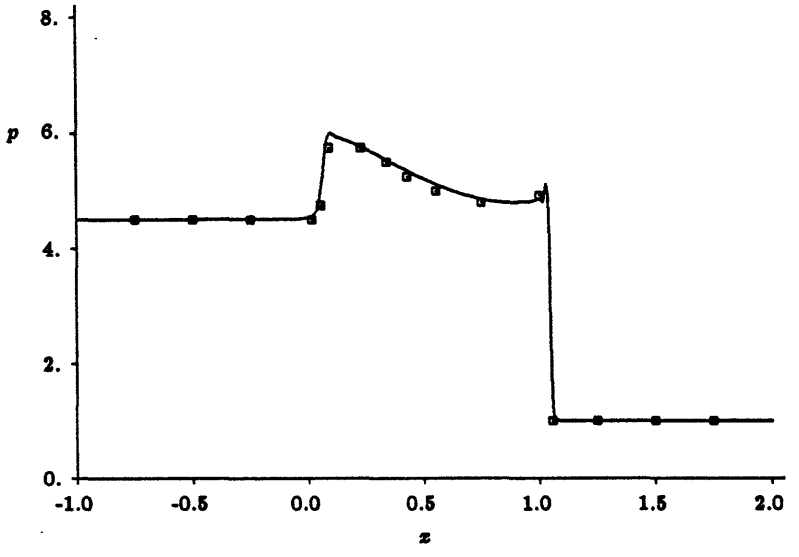

Figure 8.27: Comparison of pressure distribution at $y = 0.5$ for frozen flow over 15 % circular arc, symbols represent Yang's calculations, Reference [144].

example. The base grid consisted of 16×60 cells; hence the corresponding globally fine spatial grid would have 4^3 as many cells. Four levels of temporal strides implies that there are 2^4 smallest time-steps in each time-stride. Hence the globally fine grid in both space and time is expected to consume 512 times more CPU time than the globally coarse grid if temporal adaptation is not used in either case.

The grid is uniformly pre-embedded by six cells at $t = 0$ on both sides of the shock as shown in Figure (8.18). The unnecessarily fine region generated by pre-embedding to the left of the initial shock reverts back to the coarse grid as soon as the *usual* spatial adaptation process is turned on, as is evident in Figure (8.19) by the grid pattern at $t = 0.2$. Density was used as the refinement criterion with $R_{d1} = 1.2$ and $C_{f_d} = 0.2$ (see Section 5.3.3). Spatial adaptation was performed after each time-stride unit and the spatially adapted grid was indiscriminately extended by two additional cells on each side.

At about $t = 0.2$ the shock reaches the leading edge of the bump and soon after a compression wave ensues from the bump which propagates upstream. The compression region strengthens and develops into a shock wave which propagates against the flow stream. The Mach number in the inlet region following the initial shock is $M_i = 0.96$ or in terms of velocity $u_i = 1.48$. The velocity of the lower leg of the shock moving against the stream is (from the density contours) $u = -0.36$, or $M = 1.20$ in a frame of reference attached to the lower leg and the inlet sound speed.

At $t = 0.4$, Figure (8.20), the frontal shock has traversed about 50% of the bump. The shock wave ensuing from the bump itself has interacted with the frontal shock to form a lambda shock structure. The slip line emanating from the triple point is apparent both from the density contours and the embedded grids. It is also observed that the triple point moves vertically upwards as the frontal shock moves downstream.

As shown in Figure (8.20), at $t = 0.6$ the lower leg of the frontal shock is about to leave the bump. The triple point continues to move upward primarily due to the transverse motion of the upstream facing shock. The slip line does weaken due to the interaction with the expansion emanating from the rearward face of the cascade, which

also has the effect of distorting the triple point itself. At a still later time the reverse moving shock reaches the top wall and its reflection further degrades the triple point. Eventually the slip line decays due to the expansion from the lower wall and the influence of the reflected shock.

Corresponding to $t = 0.8$, Figure (8.22) shows the frontal shock downstream of the bump surface, and the rearward facing shock developing a strong reflection. Still another strong shock has developed at the trailing edge of the bump which interacts with the lower part of the frontal shock.

Figure (8.23) shows the situation at $t = 1.0$. The frontal shock has divorced itself from the wall interactions of the channel. The reflected shock from the upper surface continues to get stronger. The shock at the trailing edge has also begun to move upstream.

Figure (8.24) shows the situation at $t = 1.2$ after the frontal shock has left the computational domain. Non-reflective boundary condition has been applied at the exit. The reflected bow shock of the upstream facing front and the shock which originated from the trailing edge are now strengthening and moving upstream.

A similar frozen case has been studied by Yang *et. al.* [144]. Figure (8.25) compares their pressure contours for a globally fine grid at a time when the lower leg of the frontal shock is just at the bump trailing edge. This corresponds to $t = 0.65$ for the present case. Figures (8.26) and (8.27) indicate comparisons on the lower channel wall and at $y = 0.5$, the symbols represent Yang's calculations and the data has been interpolated from Figure (8.25). The agreement between the two solutions is quite reasonable.

Figure (8.28) shows the density distributions along the lower channel wall at various time-stations. The vertical scale corresponds to the initial condition $t = 0$ and all other curves are displaced by a vertical offset of 0.8. These curves are also indicative of the chronicle which has already been explained.

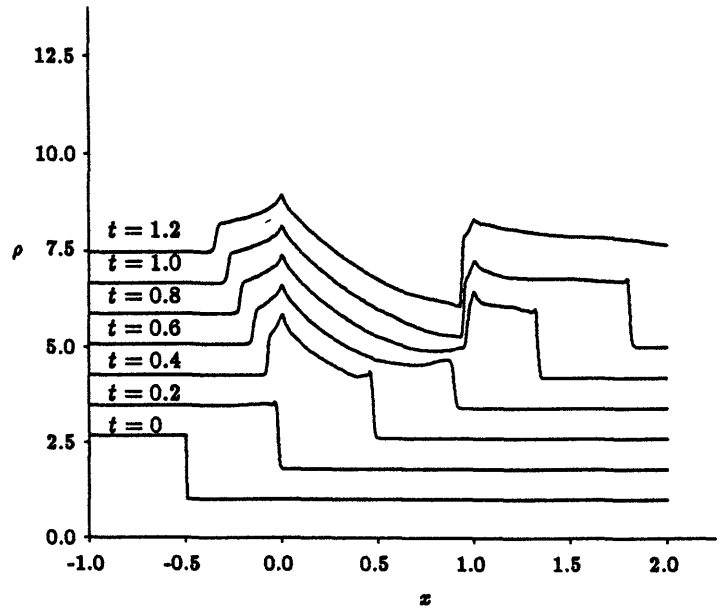

Figure 8.28: Density profiles at the lower channel wall for frozen flow over 15 % circular arc, $M_\infty = 2$.

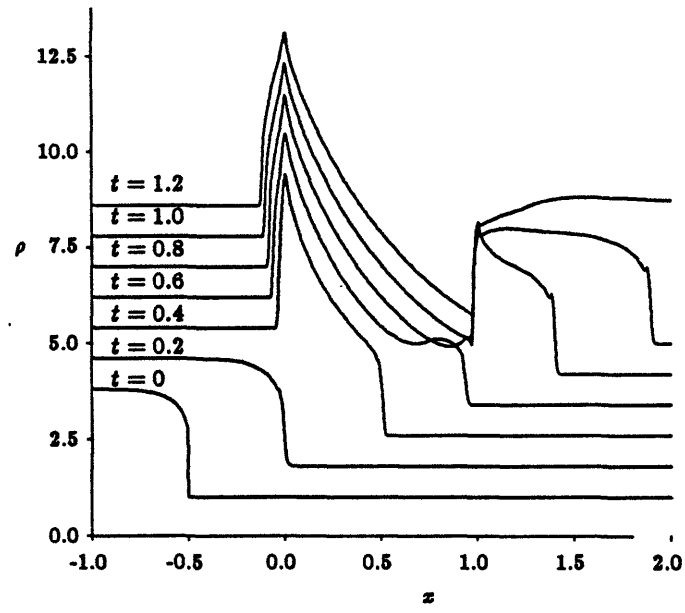

Figure 8.29: Density profiles at the lower channel wall for dissociating flow over 15 % circular arc, $M_\infty = 2$.

8.2.2 Reacting Bump Case

A second example uses a Lighthill dissociating gas flowing over the same 15% circular arc bump. The constants for the Lighthill model for oxygen are

$$\eta = 0, \quad \theta_D = 59500K, \quad \rho_D = 150 \times 10^3 \ kg/m^3. \tag{8.8}$$

Temperature and density at the inlet have been chosen so as to yield 40% dissociated oxygen atoms under equilibrium conditions. The conditions at inlet and exit are shown in Table (8.2). This corresponds to a shock moving through the channel at $M_f = 2$. The exit conditions are also the reference values for both the frozen and reacting cases. Note that the degree of dissociation at the exit corresponds to $\gamma = 1.417$ and hence a comparison with the previous frozen case can be made. Although the shock Mach number is the same in the two cases, the temperature, density and pressure ratios are very different. These ratios are also shown in Table (8.2). A choice of reaction parameter $\Phi = 10^4$ implies the relaxation length to be $x_t = 0.273$ times chord length.

The initial distributions of density and atom mass-fraction is apparent in Figures (8.29) and (8.30) which correspond to $t = 0$ curves. The relaxation following the leading shock is clearly evident in the initial field. As shown in Figure (8.31) the pre-embedded grid at $t = 0$ spans a larger domain due to the gradients in the relaxation zone trailing the frontal shock. The base grid is again composed of 16×60 cells with allowance for three spatial refinements and advancement by five temporal stages. Temporal resolution was based upon mass fraction of dissociated atoms. Spatial adaptation was performed after each time-stride unit and the spatially adapted grid was extended by two cells on each side. The refinement parameter was based on density and mass fraction of atoms with $R_{d1} = 1.2$ and $C_{f_d} = 0.2$.

Figure (8.32) shows the density contours and the associated spatial grid at $t = 0.6$ [compare with Fig. 8.21 for frozen case]. The embedded grid is again seen to be capturing the salient features of the flow field. The motivation for showing spatial grids along with line contours has been to demonstrate the grid tracking capability for all necessary features without the introduction of spurious oscillations. Since this objective has been

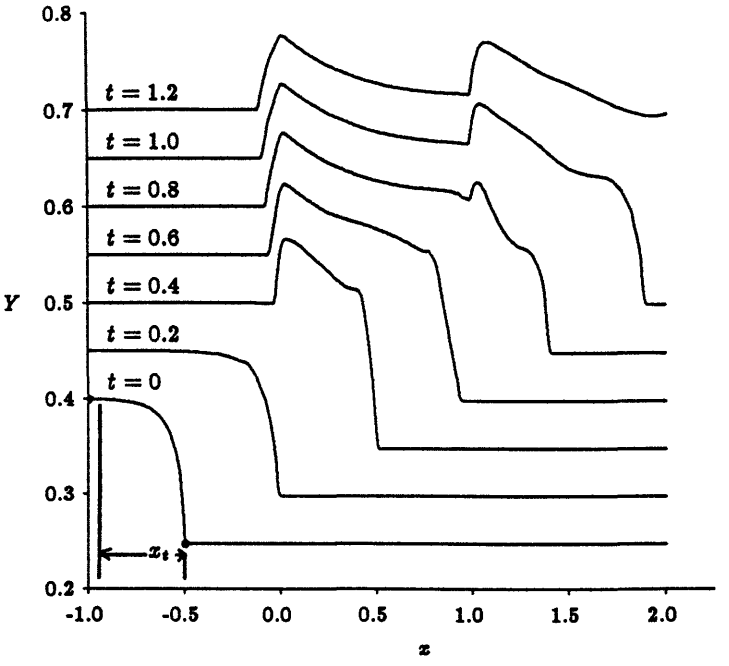

Figure 8.30: Atom mass fraction distributions at the lower channel wall for 15 % circular arc, $M_s = 2$.

	T, K	ρ, kg/m^3	Y
inlet, $()_i$	5000	3.820	0.400
exit, $()_e$	4125	1.004	0.247

gas	T_i/T_e	ρ_i/ρ_e	p_i/p_e
reacting	1.212	3.803	5.174
frozen	1.688	2.667	4.500

Table 8.2: Initial values for circular arc bump case.

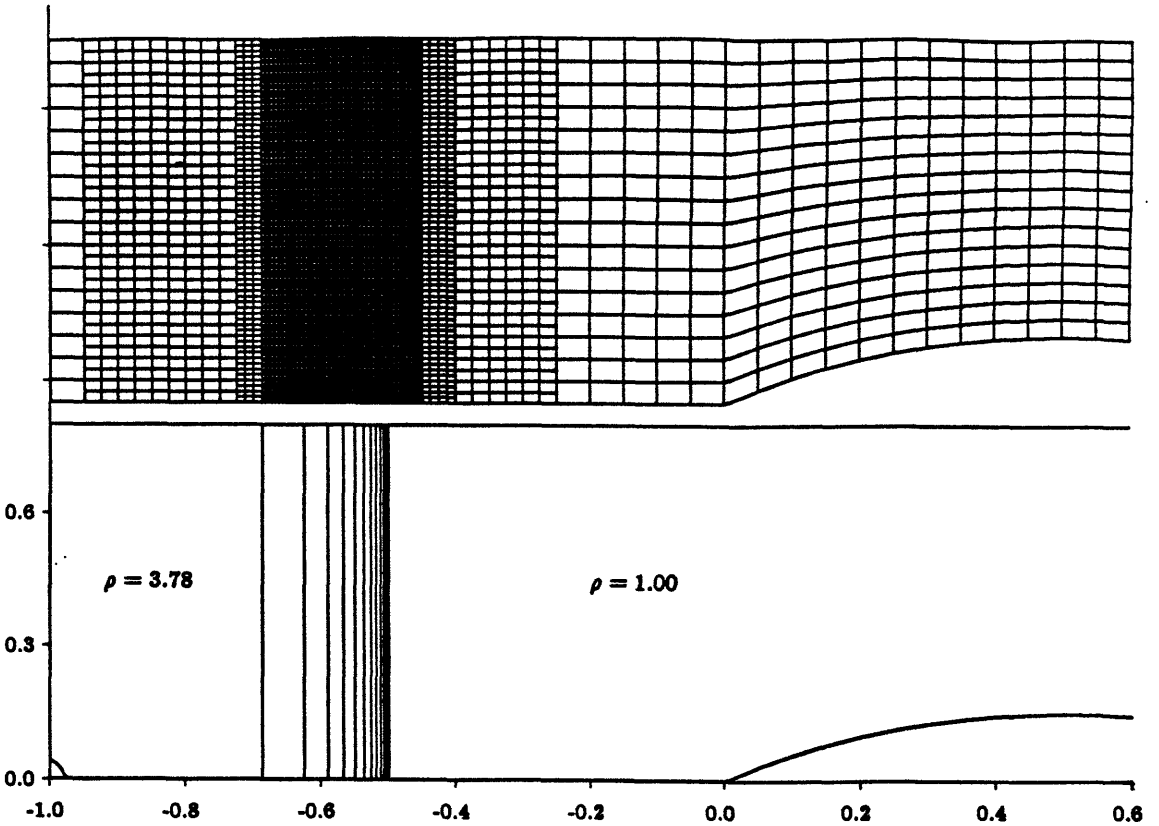

Figure 8.31: Grid and density contours at $t = 0$ for dissociating flow over 15 % circular arc bump, $M_f = 2$.

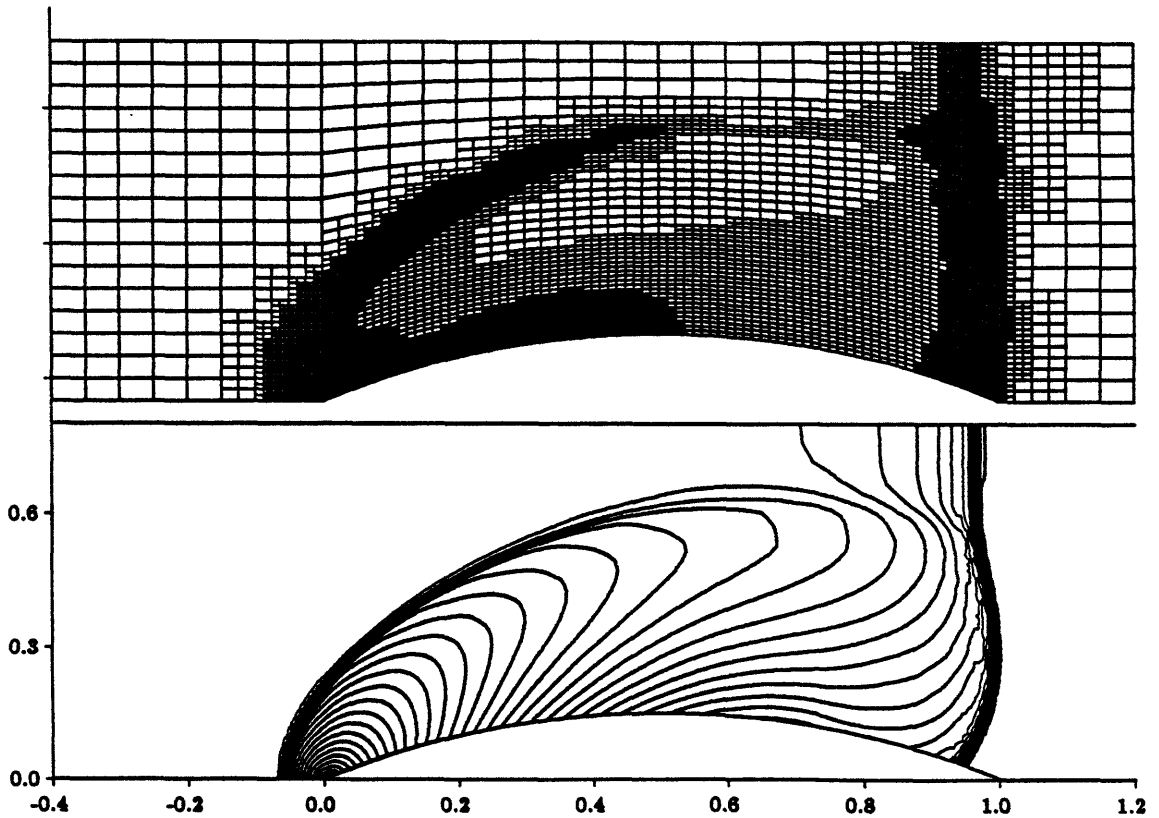

Figure 8.32: Grid and density contours at $t = 0.6$ for dissociating flow over 15 % circular arc bump, $M_f = 2$.

achieved to some extent, most spatial grids are omitted in the remaining part of the chapter. If only density is used as a refinement parameter, the spatial adaptation fails to resolve the relaxation tail. However, a combination of density and atom mass fraction yields satisfactory resolution of both frontal shock and the relaxation zone. Figure (8.33) shows the distribution of the spatial variations (Eq. 5.4) of density and atom mass fraction for the spatial grid at $t = 0.6$ on *standardized scales* which allow for unbiased spread of data. The numerical values of the averages μ_j were about seven orders of magnitude smaller than the (diagonal) standard deviations. Each square on the figure represents a single cell in the domain which number to $N = 8508$ at that time. The cells with large variations correspond to the data outside the *divide threshold ellipse* and are the cells marked for possible division. The fact that the ellipse has small eccentricity implies that the correlation in between the two variations, Figure (8.33), is relatively small. The threshold ellipse corresponds to $R_d = R_{d2} = 1.8$ and is the locus of the points satisfying Equation (5.7) with $r^2 = R_d$. The cells falling within the *collapse threshold ellipse* are marked for possible merger. About 70% of the cells lying in between the two ellipses remain unaffected. Figure (8.34) shows the corresponding cumulative frequency versus refinement parameter (the two variations are lumped together by Eq. 5.7); the threshold $R_{d2} = 1.8$ is clearly seen to correspond to 20% cells falling above this limit. It is appropriate to emphasize that the histogram records for each cell are updated whenever spatial adaptation is desired, and this procedure is done automatically as the solution evolves.

The contours of density and mass fraction of atoms are shown in Figures (8.35) and (8.36) at various time stations. The time history of this case can also be examined by observing the distributions of density and atom mass fraction along the lower channel wall as shown in Figures (8.29) and (8.30). The offset for Figure (8.29) is 0.8; for Figure (8.30) it is 0.05.

At $t = 0.2$, the frontal shock reaches the leading edge of the bump and the relaxation tail still remains unaffected. At $t = 0.4$ the lower leg of the frontal shock has traversed about 50% of the bump. The shock wave ensuing from the bump interacts with the frontal shock and the relaxation tail and forms a complex triple point. The tail becomes

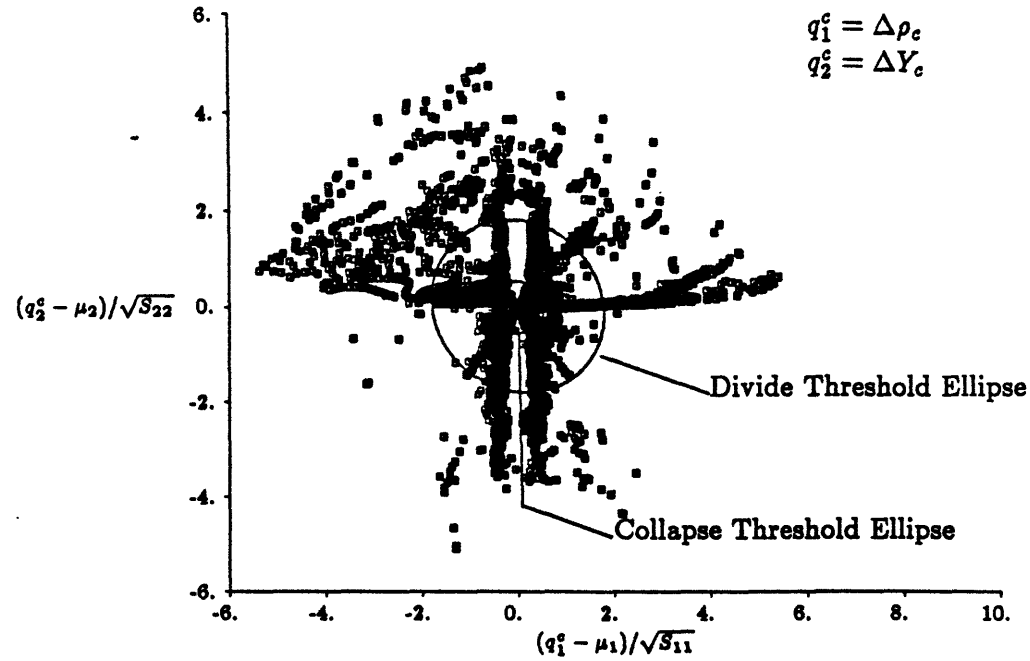

Figure 8.33: Distribution of variations for spatial adaptation at $t = 0.6$ over a 15 % circular arc, $M_f = 2$.

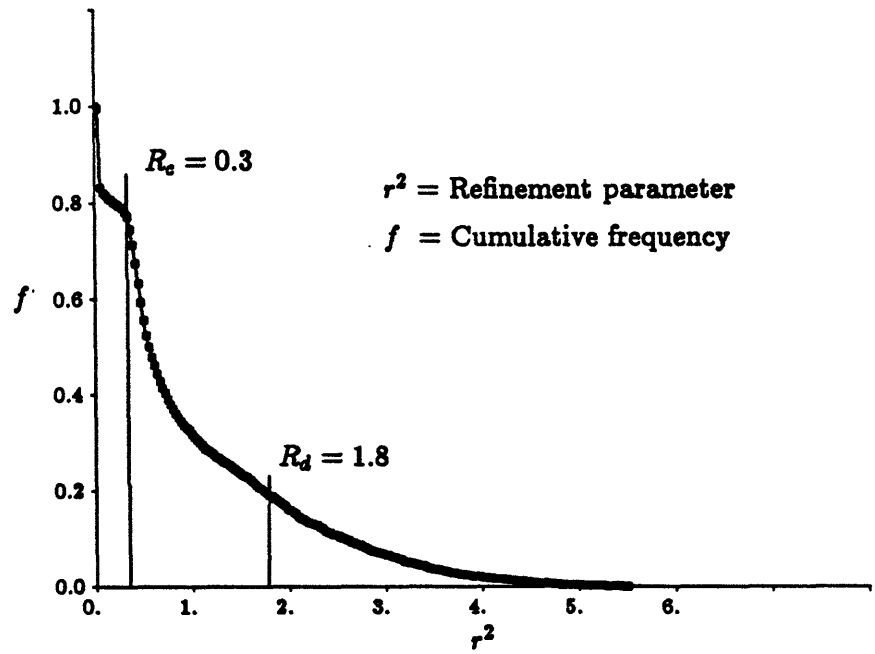

Figure 8.34: Threshold limits for spatial adaptation at $t = 0.6$ over a 15 % circular arc, $M_f = 2$.

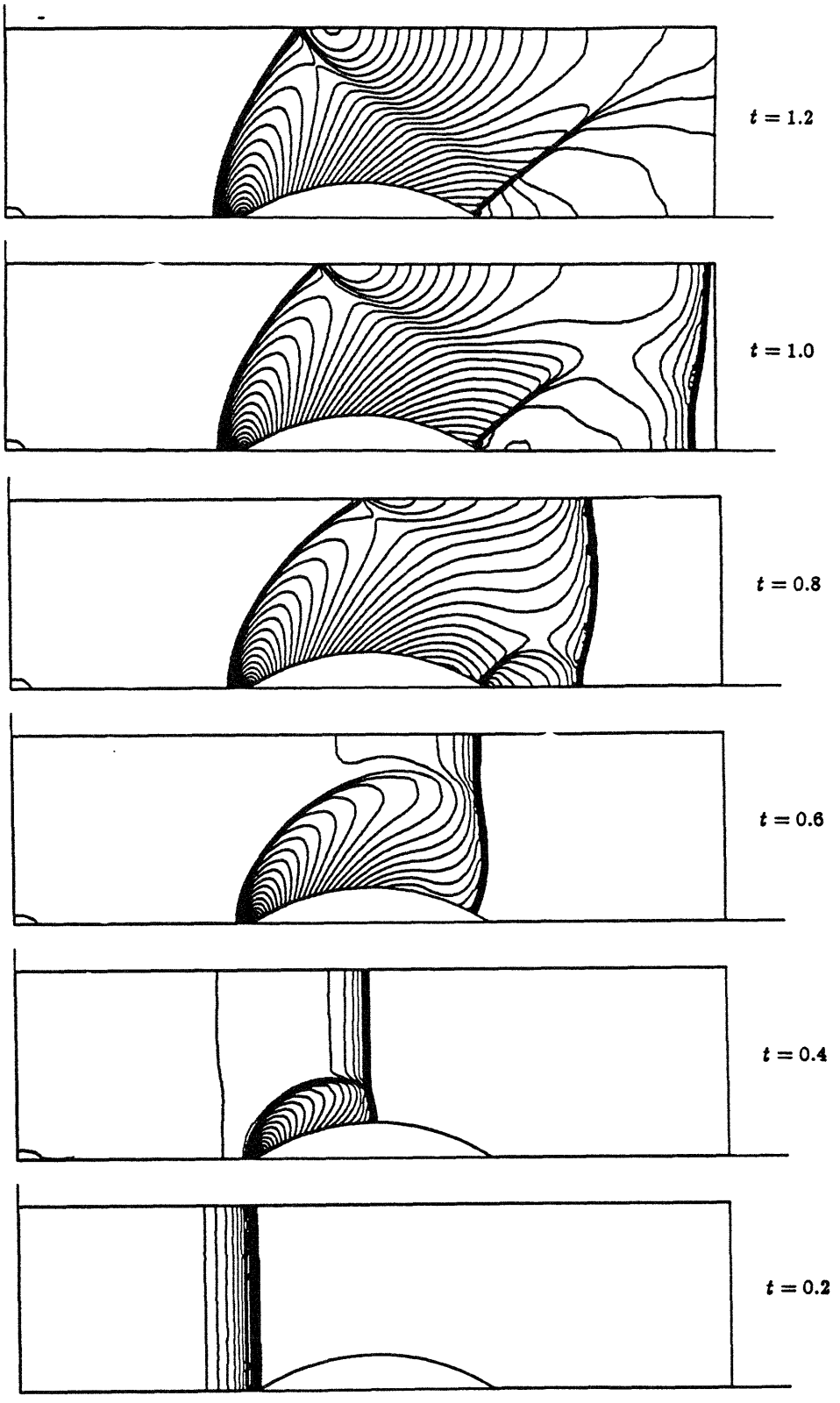

Figure 8.35: Density contours for dissociating flow over 15 % circular arc, $M_f = 2$.

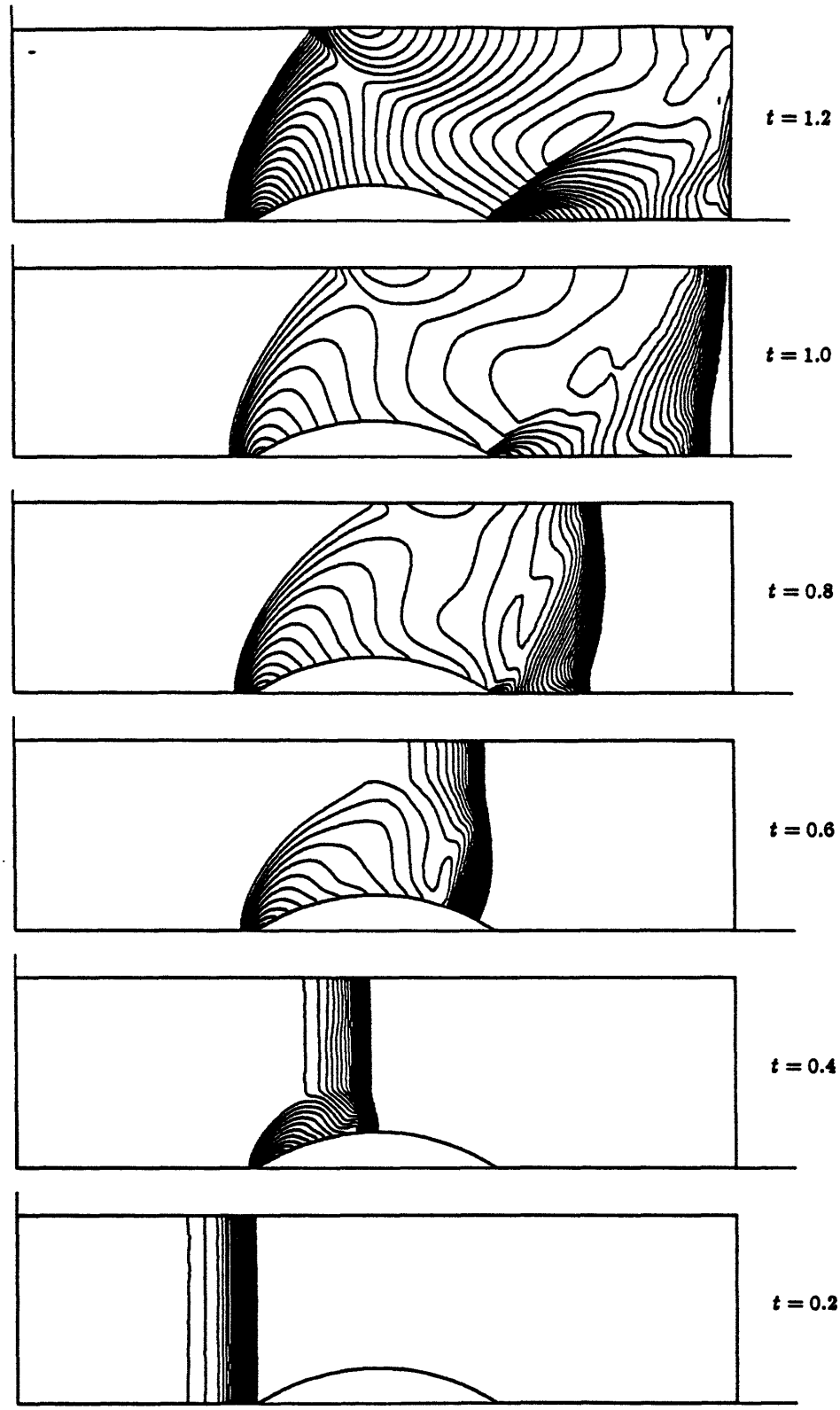

Figure 8.36: Atom mass fraction contours for dissociating flow over 15 % circular arc, $M_f = 2$.

highly distorted and small underneath the triple point. This is so since sufficient time has not elapsed after the initial interaction for a new relaxation region to emerge. The relaxation of the interaction region seems to be trapped between the slip line and the lower leg of the frontal shock. The reverse moving shock is much stronger compared to the frozen flow case which moves at a speed of $u = -0.1$ or a local frozen Mach number of 1.31 based upon the undisturbed inlet sound speed and a frame of reference attached to it. The Mach number of the inlet stream itself is $M_i = 1.24$, which is supersonic compared to the previous frozen case. At $t = 0.6$ the triple point is at about $y = 0.6$, the relaxation region is seen to be gradually increasing below the triple point, due to the longer residence time for the fluid near the bump surface. The expansion between the frontal shock and the reverse moving shock is also stronger compared to the frozen case. At $t = 0.8$, the reverse moving shock has reached the top surface and a reflection wave is developing. The relaxation region trailing the frontal shock continues to strengthen. The frontal shock has cleared the bump and the trailing edge shock is developing. At latter times the trailing edge shock remains at the same location unlike the previous frozen flow case.

8.2.3 CPU Time Comparison

In order to assess the effectiveness of the spatio-temporal adaptive algorithm, calculations have been carried out on coarse, adapted and fine grids between the time stations $t = 0$ and 0.3 for the frozen flow. In order to curtail the overhead for the globally fine grid the spatial domain was reduced to span $x \in [-0.6, 0.4]$ and $y \in [0, 0.6]$. The omitted spatial domain corresponds to regions that are either undisturbed or in the vicinity of the normal shock for $t \in [0, 0.3]$. The base grid resolution is kept the same as the previous two cases, *i.e.*, an average cell dimension of 0.05. The coarse grid corresponds to the base grid of the adapted case. Three spatial levels of embedding and four temporal stages were again allowed for the adapted case. The fine grid corresponds to the finest spatial level of the adapted case, *i.e.*, an average cell dimension of 0.00625. Both fine and coarse grid solutions were carried out with a global minimum time-step. The density contours at $t = 0.3$ for coarse, adapted and fine grids are shown in Figure (8.37).

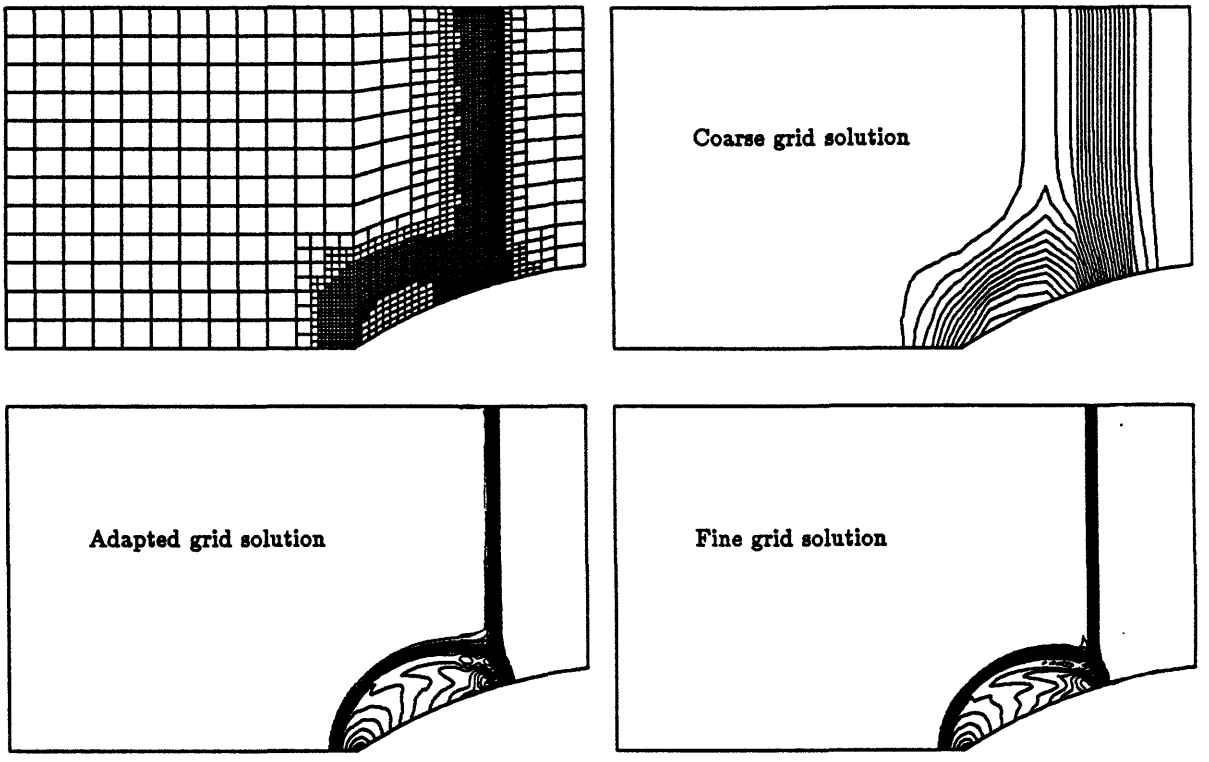

Figure 8.37: Density contours at $t = 0.3$ for coarse, adapted and fine grids for frozen flow over 15% circular arc, $M_f = 2$.

Also shown is the adapted spatial grid at that time.

The calculation on the coarse grid took a total of 51.0 seconds on a Micro-Vax II machine out of which 31.5 seconds were spent on integration. It is observed that for this coarse grid calculation initialization and output dump consume a significant fraction of the overall time. The corresponding fractions for initialization and output dump for the adapted (about 1%) and fine (about 0.2%) grid are very small. The fine grid solution took a factor of 571.3 times longer to compute compared to the coarse grid integration time (or 352.5 time for total time). The corresponding factor for the adapted grid was only 26.32 based upon integration time and a factor of 16.7 based upon the total time of the coarse grid. This spatio-temporal solution was attained 2.53 times faster compared to the one with only spatial adaptation (*i.e.*, restricted to only temporal level 0). Hence the spatio-temporal algorithm provides about an order of magnitude faster computation compared to the globally fine approach for this example. Higher adaptive levels in both space and time, especially for fast reactions, can yield up to two or three orders of magnitude faster calculations compared to the globally fine solutions. Since the fraction of the adapted grid in the previous uncurtailed domains is generally small, the savings would be larger in those cases.

It is evident that a coarse grid solution is incapable of delineating the features such as a triple point or a slip line. Furthermore the solution for the adapted grid is very close to that obtained by the globally fine grid and appears to predict the salient features at a fraction of the cost for the fine grid.

The effectiveness of the spatio-temporal adaptive algorithm increases even more when temporal resolution becomes essential in providing a prognosis for local rapid chemical adjustment. Adapted grid solutions for the dissociating case take about 7 times longer than corresponding frozen flow cases. Such reactive examples involve longer CPU time because

1. additional (species) equations are solved
2. two variables are used as detection refinement parameters in the determination of spatially resolved regions

3. temporal resolution requirements are more stringent
4. implicit integration scheme is slightly more expensive.

Since the dissociating case on a globally fine grid in space and time, even for a curtailed domain, would require prohibitively long computation, such comparisons for a reacting were not completed.

8.2.4 Frozen Duct Flow

The next example is for a frozen medium ($\gamma = 1.4$) and a shock moving at $M_f = 2.2$ through a two-dimensional 90 degree bend duct. The channel dimensions are normalized by the mean duct radius and is spanned by $x \in [-0.8, 1.2]$, $y \in [-0.5, 1.2]$. The inner and outer radii are $r_{min} = 0.8$ and $r_{max} = 1.2$ and their center is taken to be the origin of coordinates. This computation was originally carried out by Aki [3] and was subsequently repeated by Yee [145]. An experimental investigation by Takayama *et. al.* has been cited by both references. The shock is initially at $x = -0.5$. Three levels of spatial embedding beyond the base grid, and four levels of temporal strides were used here. The base grid consisted of 8 cells along the radial direction and 32 cells along the circumference of the duct, with a total of 480 cells in the domain. Spatial adaptation was performed after each time-stride and the spatially adapted grid was extended by two cells.

The density distributions along the lower and upper channel walls are shown in Figures (8.38) and (8.39). Note that the abscissa is the curvilinear distance along the respective walls starting from the inlet of the computational domain. The vertical scale again corresponds to the initial condition $t = 0$ and other curves are displaced by a vertical offset of 0.8. The density contours at various time-stations are shown in Figure (8.40).

At about $t = 0.2$ the shock reaches the bend. Soon after an expansion ensues from the lower surface and a compression initiates from the upper surface. At $t = 0.4$ the compression has strengthened and it has started interacting with the frontal shock and

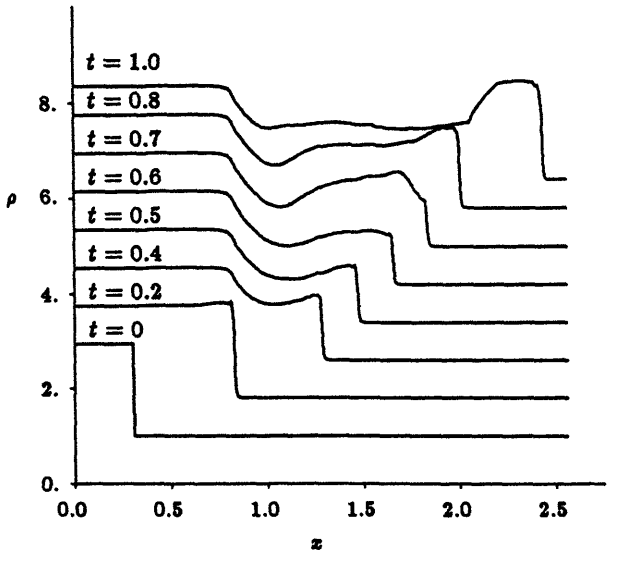

Figure 8.38: Density distributions along lower duct wall for frozen flow.

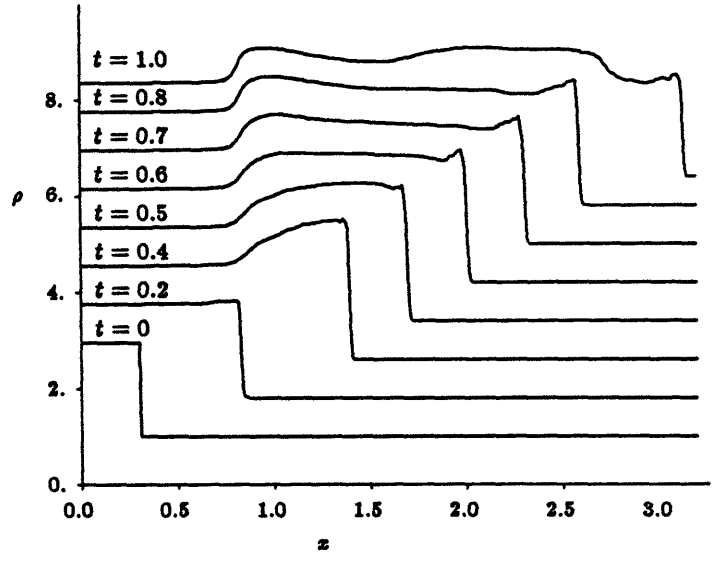

Figure 8.39: Density distributions along upper duct wall for frozen flow.

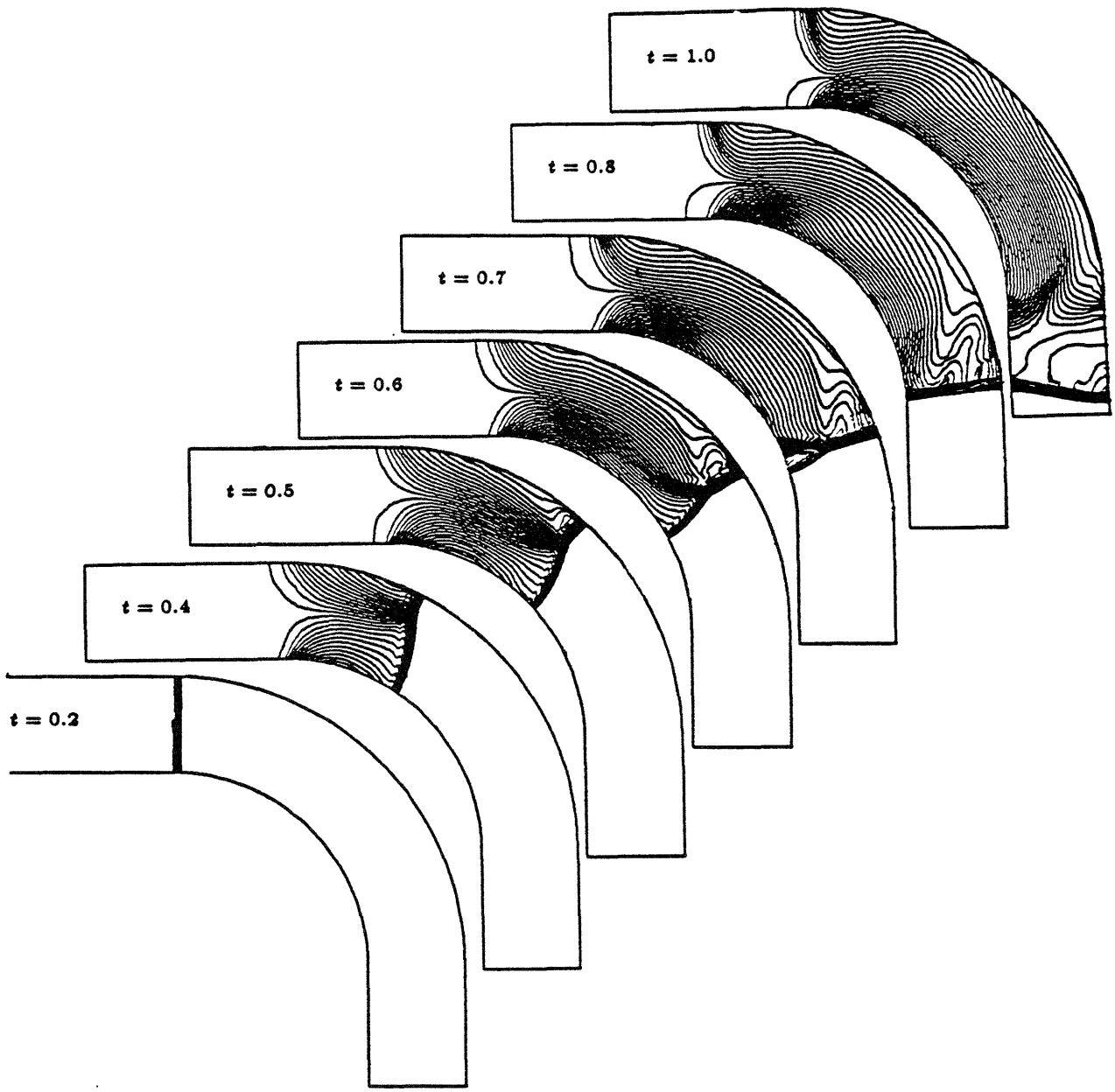

Figure 8.40: Density contours for frozen flow in bent duct, $M_f = 2.2$.

a lambda shock is about to form. At $t = 0.5$ the slip line is clearly evident, the triple point is shifting from the upper wall towards the lower wall. The compression wave has started interacting with the expansion fan and as a result the expansion is restricted to a small region hugging the lower wall. At $t = 0.6$ the same trends continue. At $t = 0.7$ a distinct lambda shock is formed, the compression has reached the lower wall and the expansion is constrained to the inlet region near the lower wall. At $t = 0.8$ the triple point has reached the lower surface, the domain of expansion is further limited, the compression at the lower wall has begun to strengthen further and latter develops into a shock. The compression at the upper inlet wall region has gradually strengthened. At $t = 1.0$ the frontal shock has managed to recover its planer structure and has divorced itself from the interactions appearing from the two curved surfaces, leaving behind a shock wave in its wake at the lower wall.

Figure (8.41) shows the density contours for this frozen case as calculated by Aki [4] by a total variation diminishing (TVD) scheme. The approximate time-levels shown here were interpolated from the location of the frontal shock. Aki had used a 176×360 grid for the curved channel which would be one level finer compared to the finest spatial level in the current calculation. It is observed that there are subtle differences in the two results. The slip line is not as sharp in the present calculation and weak reflections are not observed. These differences are also apparent in Figure (8.42) that compares the density variation of the two computations at $t = 0.6$ at the upper and lower channel walls of the bend duct. Figure (8.43) shows the infinite fringe interferogram from Reference [3] which approximately correspond to the times $t = 0.5$ and 0.7 for the current case. The calculations compare reasonably with the experiment.

8.2.5 Reacting Duct Case

The same bend duct was also considered using a Lighthill dissociating gas. The base grid was identical to the previous frozen flow case and the inlet conditions the same as the reacting bump case. However, the exit conditions for $M_f = 2.2$ are different as indicated in Table (8.3). The choice $\Phi = 10^4$ implies $x_t = 0.223$. The contours of density

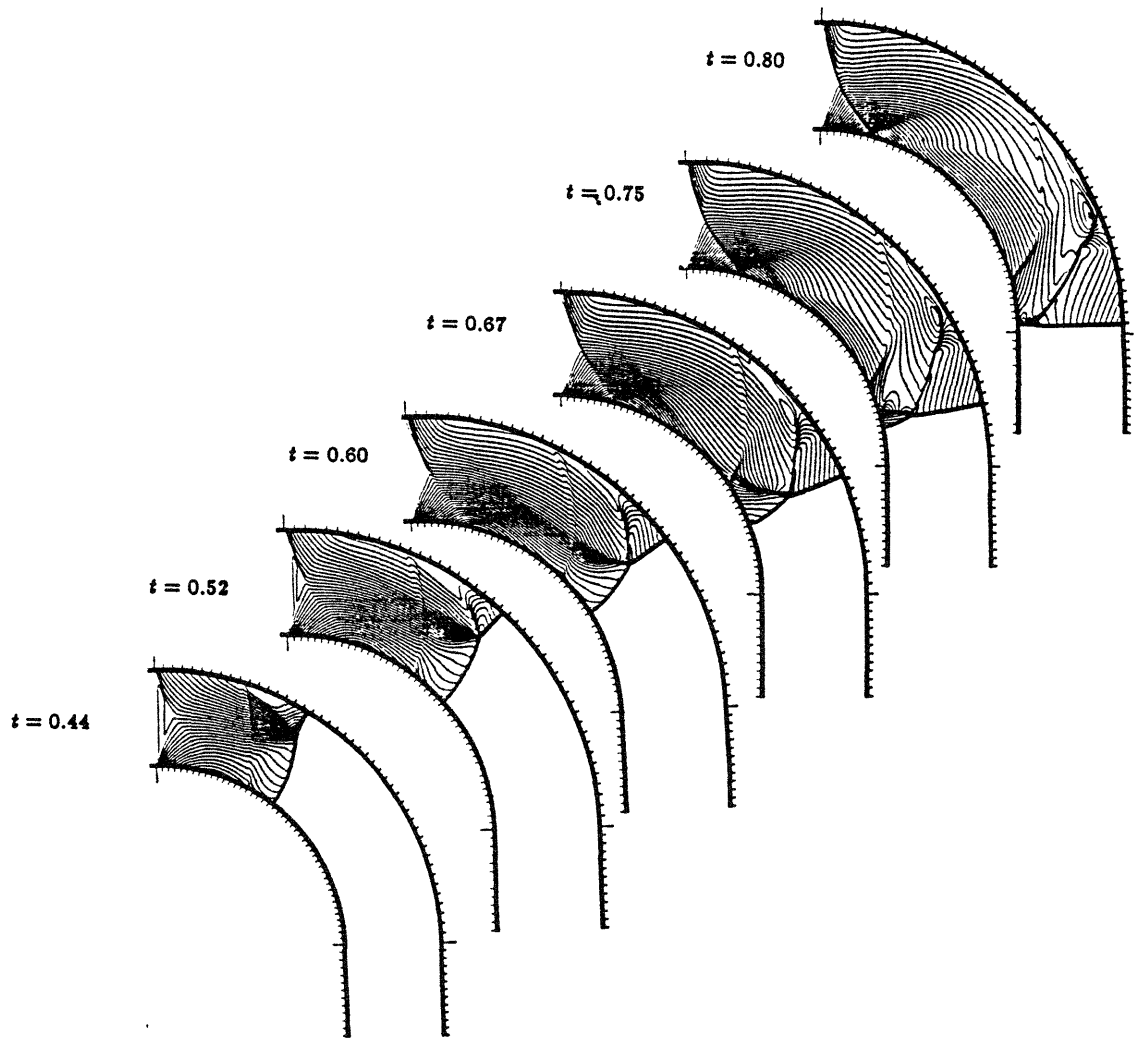

Figure 8.41: Density contours for frozen flow, $M_f = 2.2$, Aki's calculations, Reference [4].

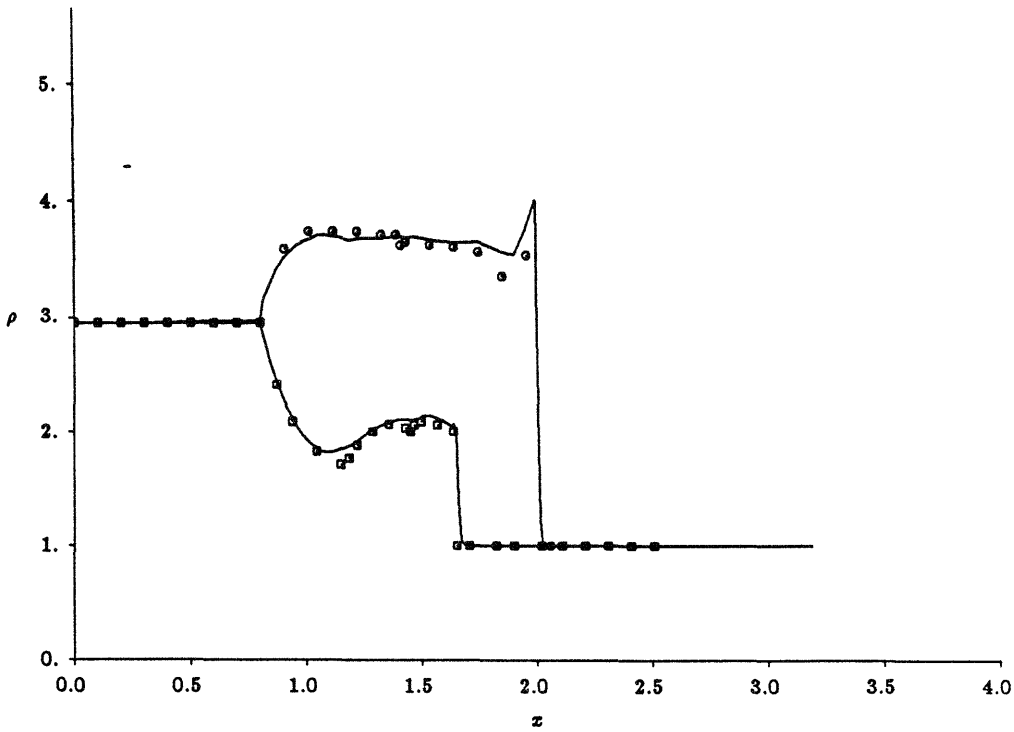

Figure 8.42: Density variations at $t = 0.6$ at upper and lower channel walls for frozen flow, $M_f = 2.2$, symbols indicate Aki's calculations, Reference [4].

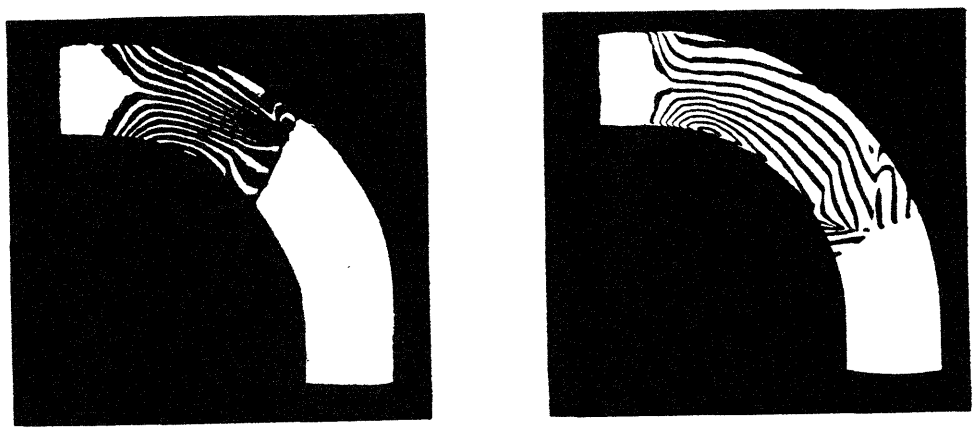

Figure 8.43: Infinite fringe interferogram for frozen flow in bend duct, Reference [4].

	T, K	ρ, kg/m^3	Y
inlet, $()_i$	5000	3.820	0.400
exit, $()_e$	4011	0.874	0.220

gas	T_i/T_e	ρ_i/ρ_e	p_i/p_e
reacting	1.247	4.369	6.250
frozen	1.857	2.951	5.480

Table 8.3: Initial values for bend case.

and atom mass fraction are shown in Figures (8.44) and (8.45). The distributions of density and atom mass fraction along the lower and upper channel walls are shown in Figures (8.46) to (8.49).

At $t = 0.4$ a complex triple point is forming, the compression and expansion fans ensuing from the upper and lower channel walls have started interacting with the relaxation tail and the frontal shock. At $t = 0.5$ the lambda shock is clearly apparent, and at $t = 0.6$ the compression from the top surface has strengthened to form a shock wave which is about to reach the lower wall. Such a strong shock at the inlet was not observed for the frozen flow. At $t = 0.7$ the trailing leg of the lambda shock is about to reach the lower surface. The shock appearing at the inlet has reached the lower surface and a reflection wave is forming. At $t = 0.8$ this reflected shock has further strengthened. The lambda shock has begun to collapse as it enters the straight portion of the duct. The slip line emerging from the triple point is affected by the relaxation tail from the beginning of its formation. It is interesting to note that the atom mass fraction remains nearly unaffected through the expansion behind the frontal shock and near the lower wall, very much like *freezing out*. The compression region near the upper wall does not show this behavior.

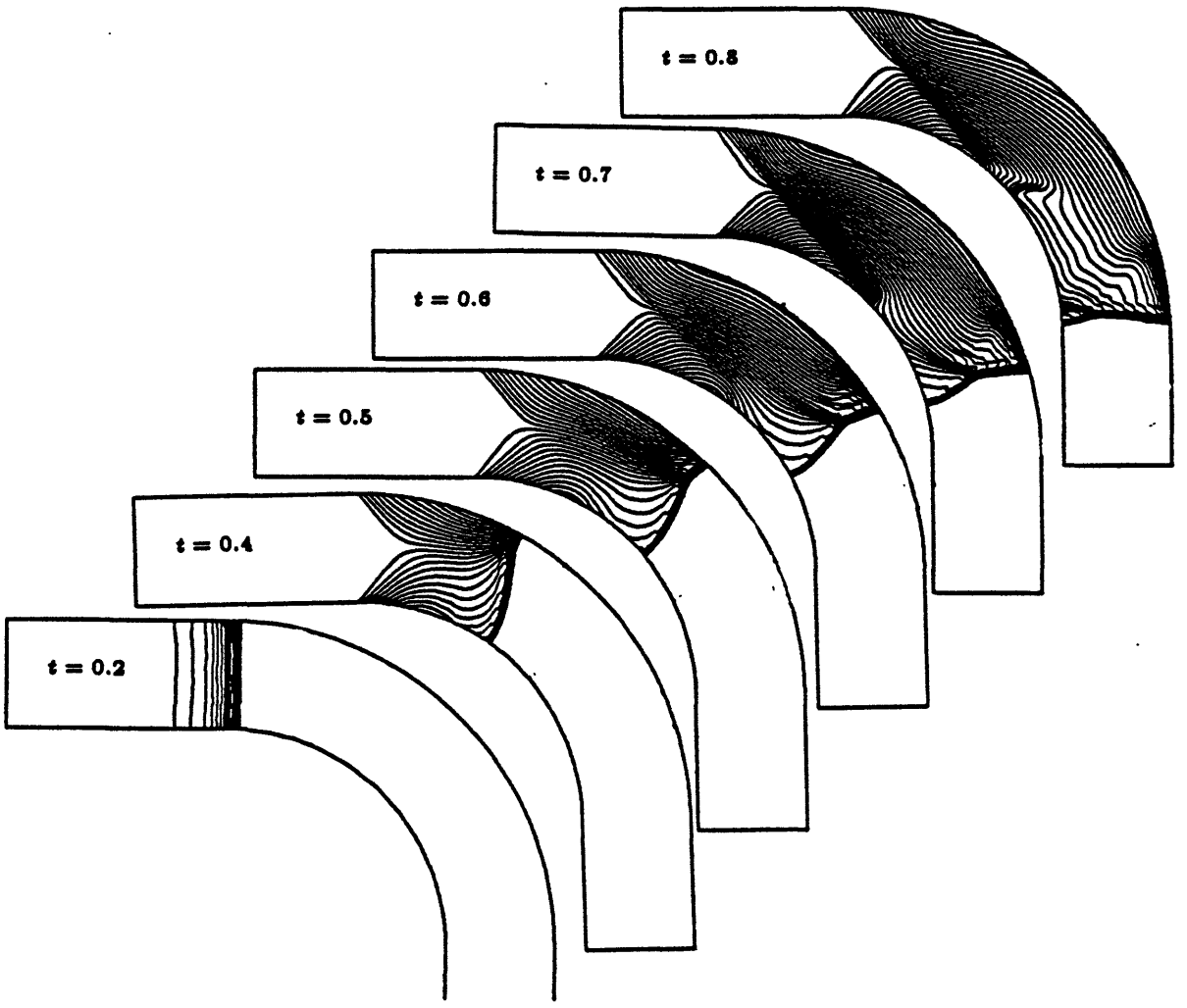

Figure 8.44: Density contours for dissociating flow in bend duct, $M_f = 2.2$.

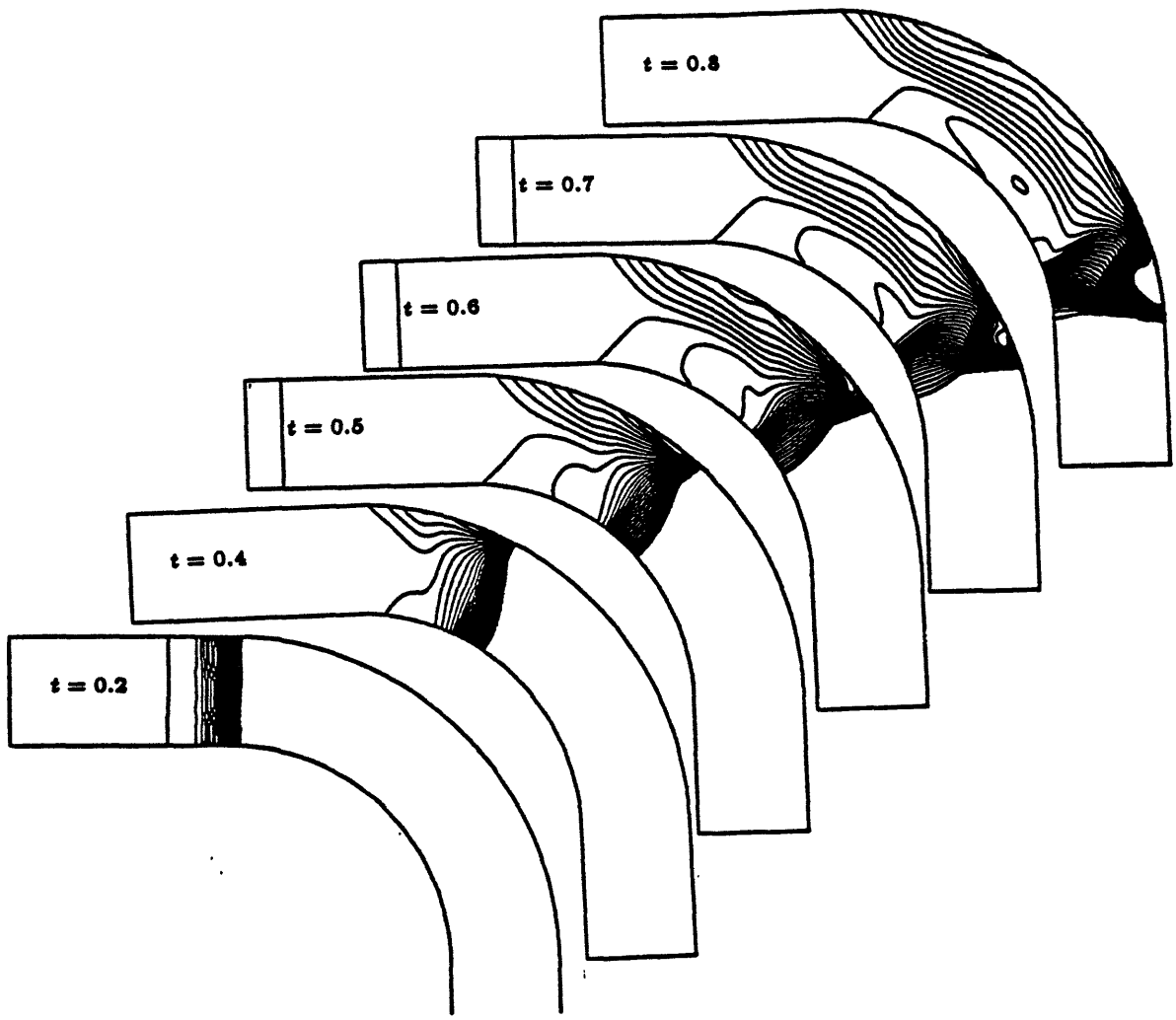

Figure 8.45: Atom mass fraction contours for dissociating flow in bend duct, $M_f = 2.2$.

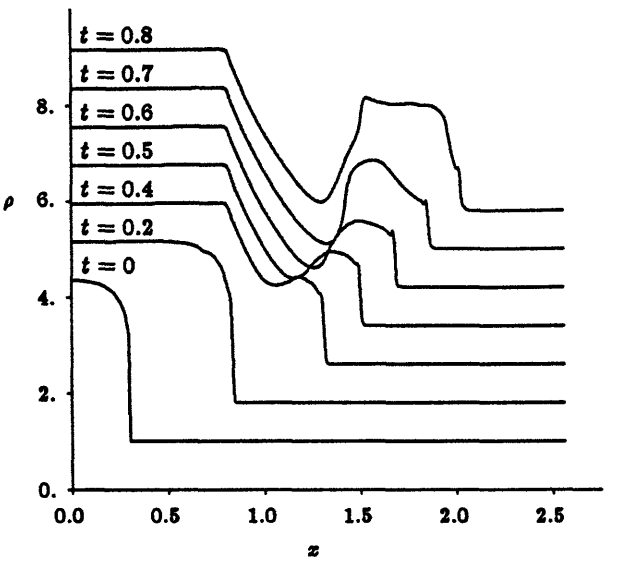

Figure 8.46: Density distributions along lower duct wall for dissociating flow.

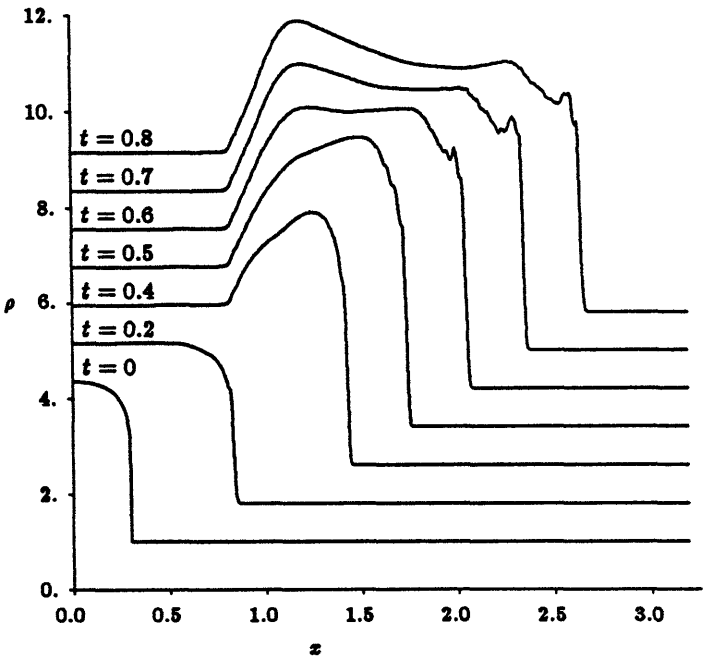

Figure 8.47: Density distributions along upper duct wall for dissociating flow.

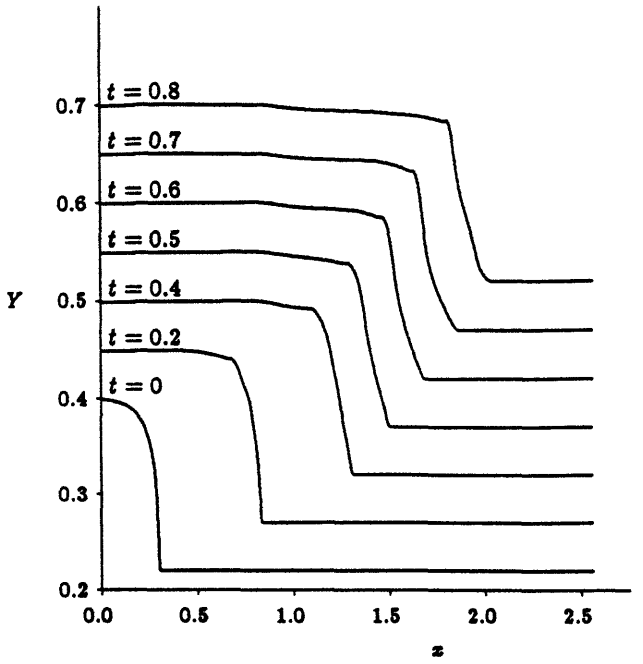

Figure 8.48: Distributions of atom mass fraction along lower duct wall for dissociating flow.

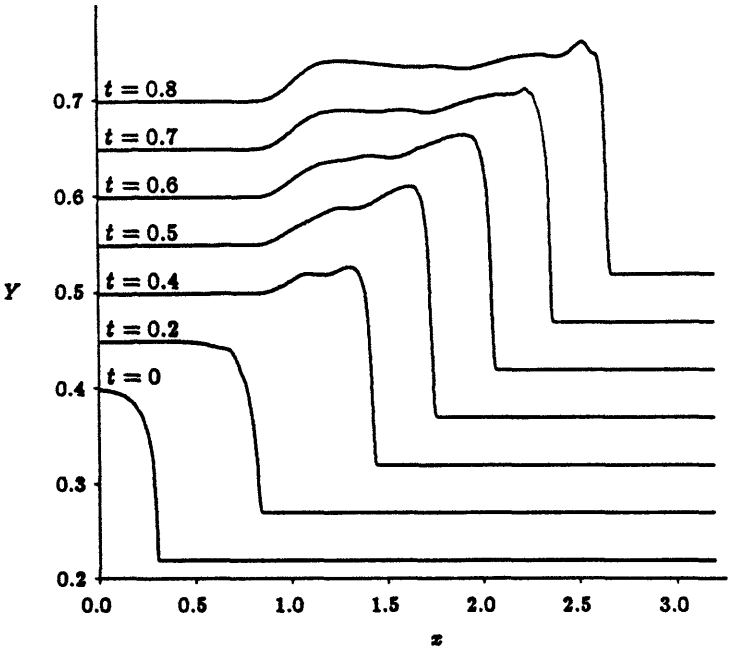

Figure 8.49: Distributions of atom mass fraction along upper duct wall for dissociating flow.

8.3 Scramjet Inlets

8.3.1 Perfect Gas Example for Two Strut Model

For the three-dimensional scramjet concept under consideration at NASA Langley, Kumar [74] has suggested a two-dimensional model that can be used to analyze the scramjet inlets. He had performed computations for a one and two strut inlet configuration using a perfect gas for flows over a range of free stream Mach numbers between 3 and 7. The calculations were performed for both inviscid and viscous models and the results indicated that Euler equations describe all the salient features of the flow field. The reference suggested the following inlet conditions for a free stream Mach number of 7.0

$$M_i = 5.03 \ , \qquad p_i = 3550Pa \ , \qquad T_i = 335K \tag{8.9}$$

for a two-strut geometry shown in Figures (8.50) to (8.52). The first of these figures establishes the labels for the scramjet inlet whereas Figure (8.51) shows the base grid, comprising of 368 cells, that was generated by the block-grid generator mentioned in Chapter 5. The external wall angles are $\alpha = 6.668$ degrees with respect to x-axis whereas the initial (leading) angles of the struts are $\beta = 11.873$ degrees as quoted by Kumar. The inner trailing wall angles of the struts are $\gamma = 7.141$ degrees whereas the external trailing angles are $\delta = 8.146$ degrees. The domain is spanned by $x \in [-0.2, 2.3]$, $y \in [-0.5, 0.5]$. The leading edges of the struts are located at $(0.6, \pm 0.2)$. The suggested reference length, for the initial channel height, is $0.15m$.

The calculation was performed by utilizing spatial adaptation while using local time-stepping by the current algorithm. Figure (8.52) shows the final adapted grid with three levels of spatial embedding beyond the base grid. Note that the third level of adaptation, near the external walls, does not extend all the way to the outer surface of the embedded struts for the choice $R_{d1} = 1.2$ and $C_{fd} = 0.2$ for spatial refinement parameter involving density differences. This example shows that the choice of threshold limits for refinement parameter is problem dependent and one has to careful in selecting the appropriate values. Lower spatial resoltion in these regions results in gradual thickening of the

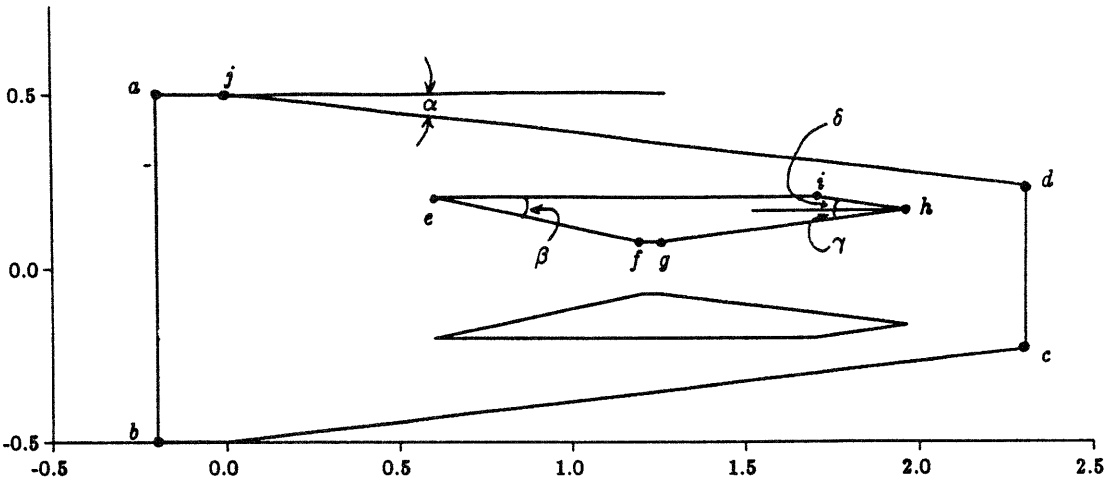

Figure 8.50: Nomenclature for two-strut scramjet inlet configuration.

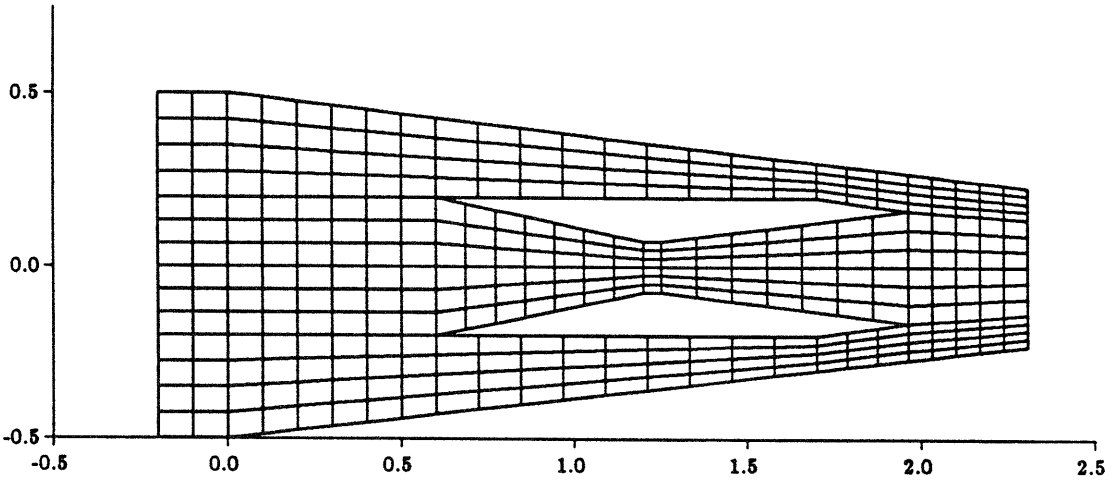

Figure 8.51: Base grid for two-strut scramjet inlet configuration.

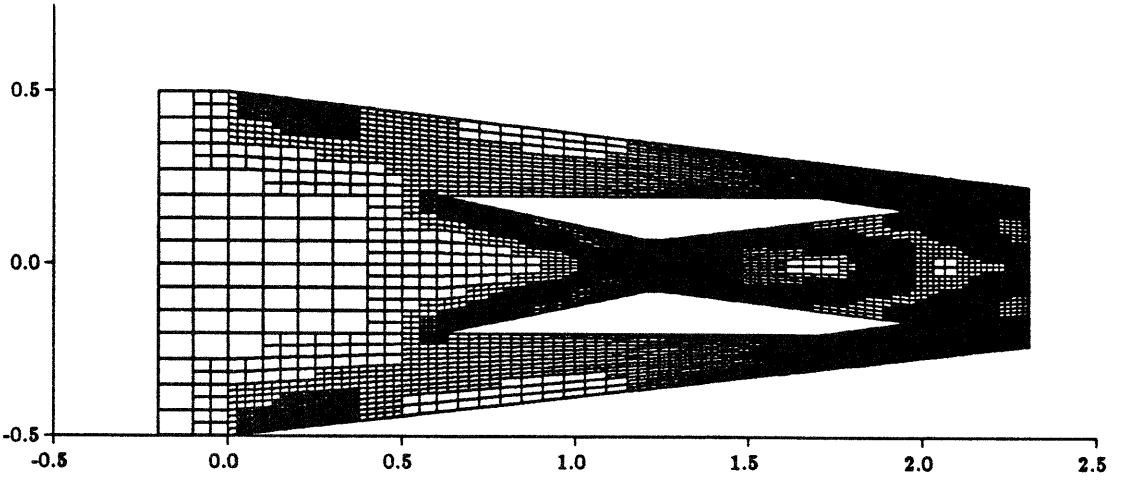

Figure 8.52: Final grid for two-strut scramjet inlet configuration, $M_i = 5.03$, perfect gas flow.

shocks emanating from the corner points j. Figure (8.53) and (8.54) show the contours of density and pressure that are generally in good agreement with the results of [74,119].

These perfect gas calculations reveal that the maximum temperature is about 2 (normalized by the inlet temperature of 335 K) and occurs between the two wedges approximately where the transverse dimension is a minimum. Since the combustion of hydrogen below about 1000 K is negligible, it does not appear that the present configuration would support significant amount of combustion on a continual basis. In order to sustain combustion with a fixed geometry the inlet temperature and pressure can be raised by increasing the inlet Mach number, or alternatively, by increasing the wedge angles while keeping the inflow conditions fixed. It has also been suggested by Martinez-Sanchez [86] that pressures in excess of nearly one atmosphere are needed for significant combustion of hydrogen in air.

8.3.2 Premixed Flow Example for Two Strut Model

Consider a flight Mach number of $M_\infty = 10$ at an altitude of 20 miles where the representative atmospheric conditions are

$$p_\infty = 1000Pa \ , \qquad T_\infty = 200K \tag{8.10}$$

Assuming two 7 degree wedges that turn the flow in the same direction and a 14 degree return produced by a cowl plate as shown in Figure (8.55); the conditions just after the third shock for a perfect gas ($\gamma = 1.4$) yield the following conditions

$$\frac{T_i}{T_\infty} = 4.64 \ (927K) \ , \qquad \frac{p_i}{p_\infty} = 81.18 \ (81180Pa) \ , \qquad M_i = 4.20 \tag{8.11}$$

Taking the variations of the changes in ratio of specific heats at high temperature for air into account such inlet conditions are approximately

$$p_i = 80000Pa \ , \qquad T_i = 880K \ , \qquad M_i = 4.30 \tag{8.12}$$

Similarly for a flight Mach number of 20 at an altitude of about 30 miles the representative conditions are

$$p_\infty = 350Pa \ , \qquad T_\infty = 300K \tag{8.13}$$

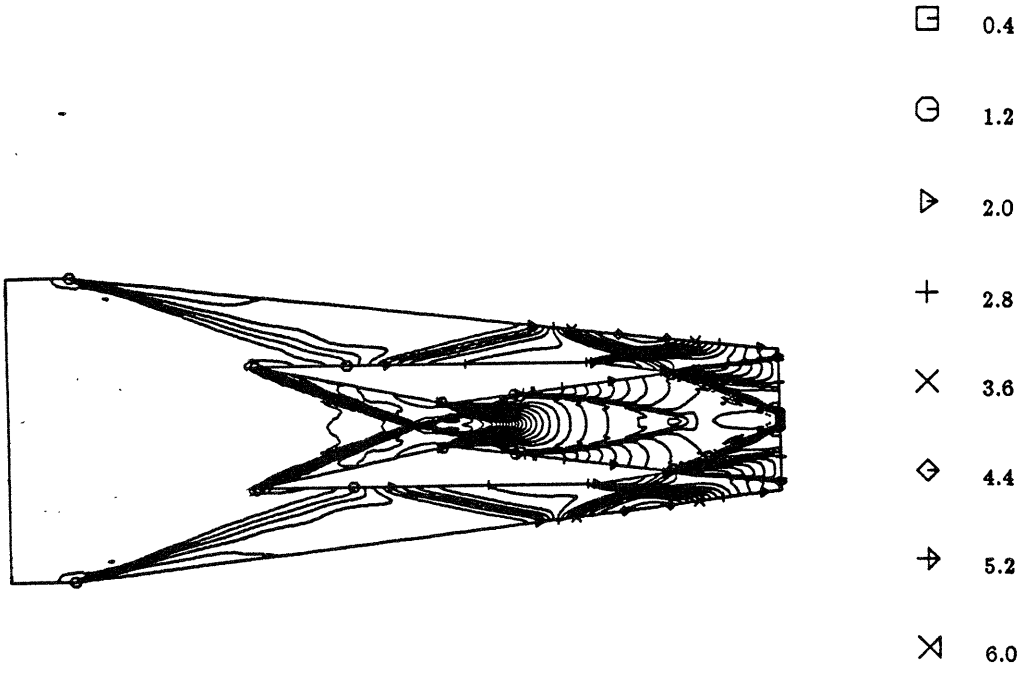

Figure 8.53: Density contours for two-strut scramjet inlet, $M_i = 5.03$, frozen flow.

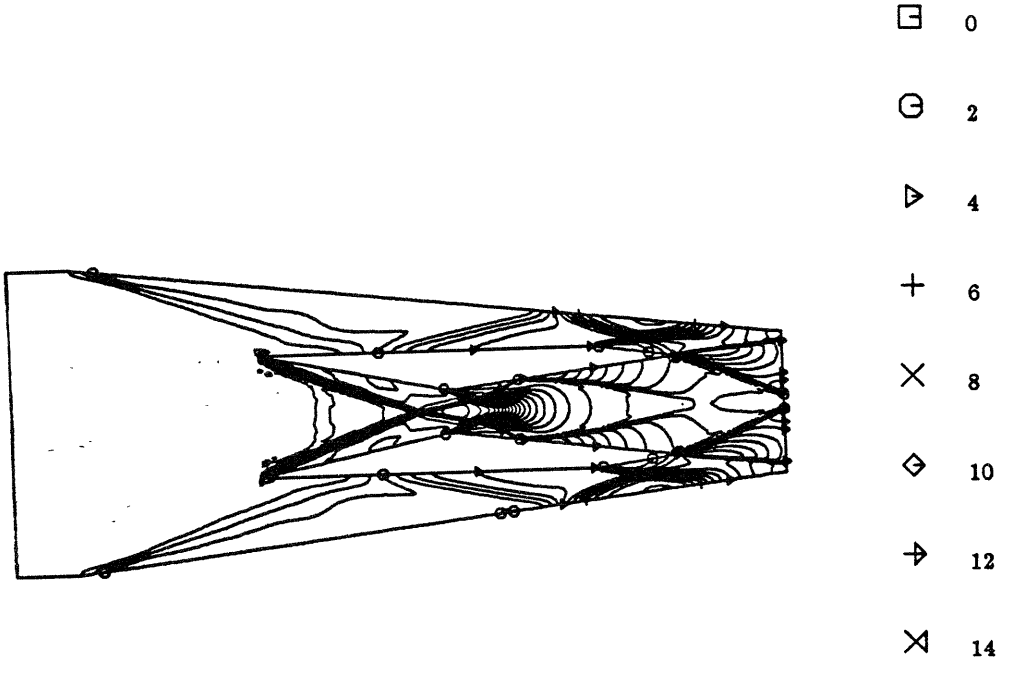

Figure 8.54: Pressure contours for two-strut scramjet inlet, $M_i = 5.03$, frozen flow.

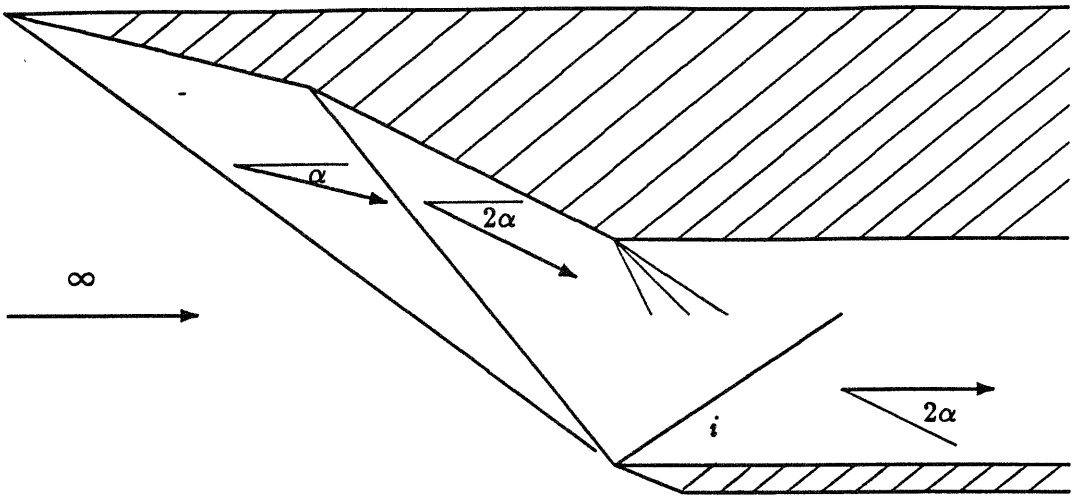

Figure 8.55: Sketch of a model scramjet configuration.

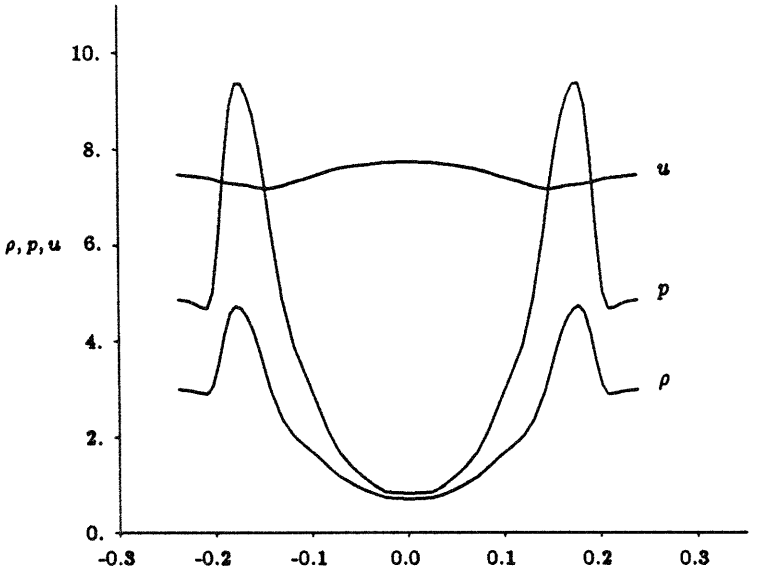

Figure 8.56: Distribution of density, pressure and velocity in the exit plane for two-strut scramjet inlet, premixed frozen flow.

For this case if the angle α in Figure (8.55) is 5 degrees, then the conditions following the third shock are

$$p_i = 80000Pa \ , \qquad T_i = 2300K \ , \qquad M_i = 6.6 \tag{8.14}$$

For this case some cooling of the incoming air may be needed if the fuel is injected ahead of the scramjet inlet. These simple calculations indicate that the mechanism of raising the pressure and/or temperature of the incoming air by inlet shocks is a viable one and it is generally possible to raise the pressures to about an atmosphere (or more) inside the region where combustion is to take place.

For the purposes of computations, the previous geometry, Figure (8.51), is assumed to follow after the third shock in between the cowl plate and the inner surface of the scramjet and the effect of the expansion fan is neglected, that is, the above flow conditions are assumed to be as uniform at inflow to the geometry. The fuel is assumed to be injected somewhere after the second leading shock in Figure (8.55) and the flow is assumed to be thoroughly mixed before it enters the computational domain. Hydrogen is assumed to have an equivalence ratio of unity and Rogers and Chinitz model of hydrogen combustion is used.

Two separate runs were carried out for this premixed fuel addition example for comparitive purposes. In the first case hydrogen was present but was not allowed to react and the finite rate chemistry was turned on in the second case. These cases were done by using the same base grid as in the previous frozen case with a total of three spatial embedding levels. The Mach number of the incoming air is 6.6 with a temperature of 800 K and pressure of 0.8 atmosphere.

Figure (8.56) shows the variation of the density, pressure and the x-component of velocity at the exit plane, $x = 2.3$, for the premixed frozen flow case. The corresponding contours of density and pressure are shown in Figures (8.57) and (8.58).

Figure (8.59) shows the variation of the density, pressure and velocity at the exit plane for the reacting gas. It was noted that the average pressure at the exit plane had increased from a frozen flow value of 4.1 to 7.2 (normalized by inlet pressure) in

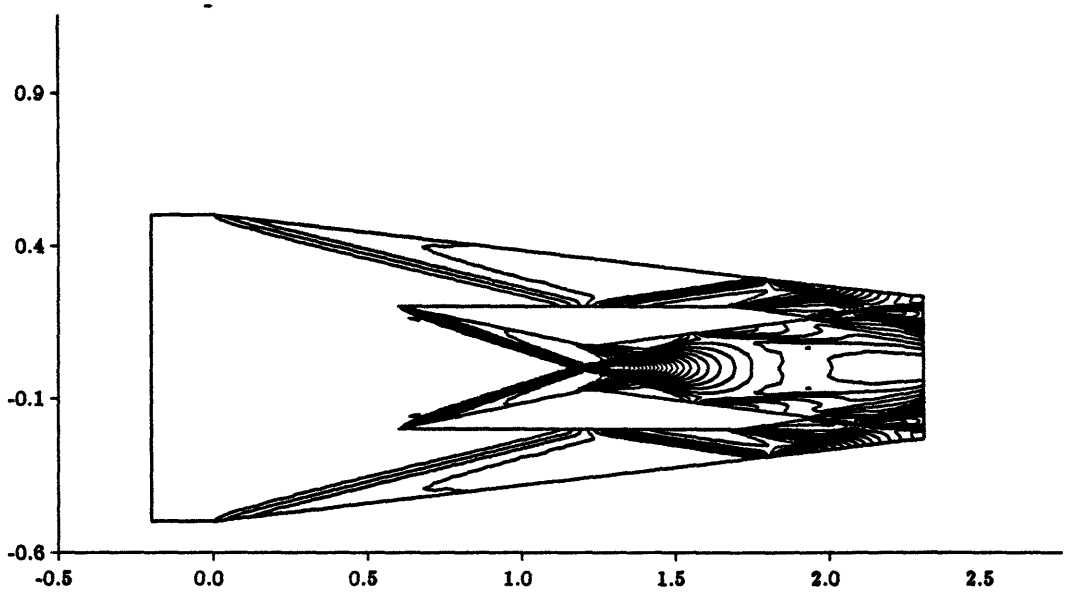

Figure 8.57: Density contours for two-strut scramjet inlet, premixed frozen flow.

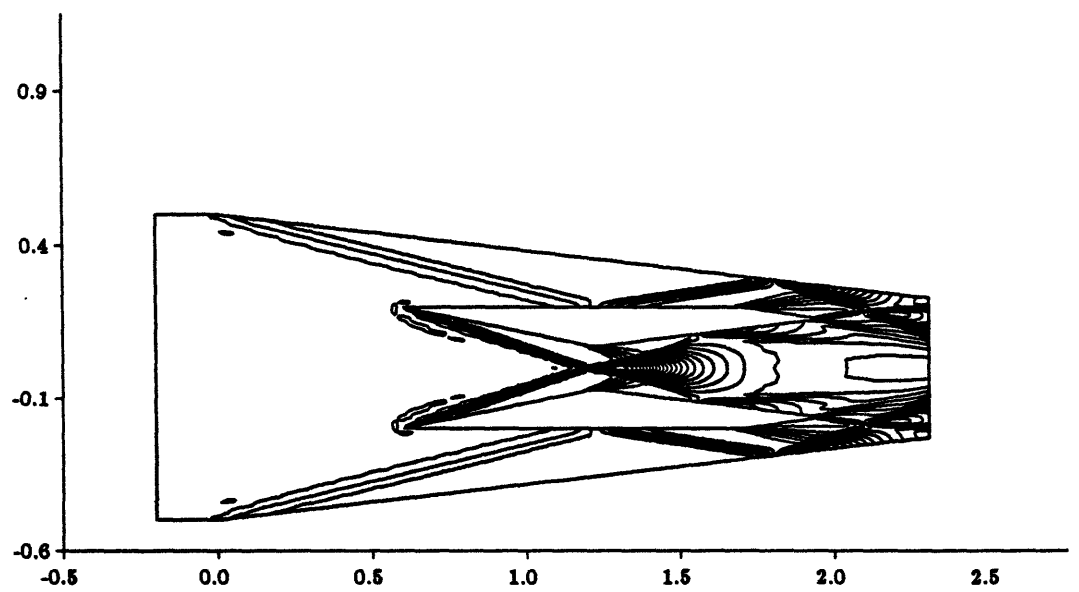

Figure 8.58: Pressure contours for two-strut scramjet inlet, premixed frozen flow.

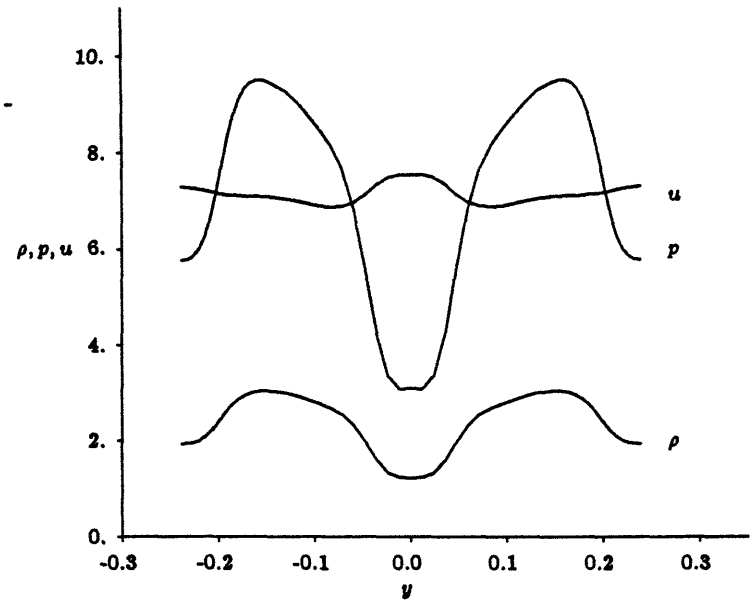

Figure 8.59: Distribution of density, pressure and velocity in the exit plane for two-strut scramjet inlet, premixed reacting flow.

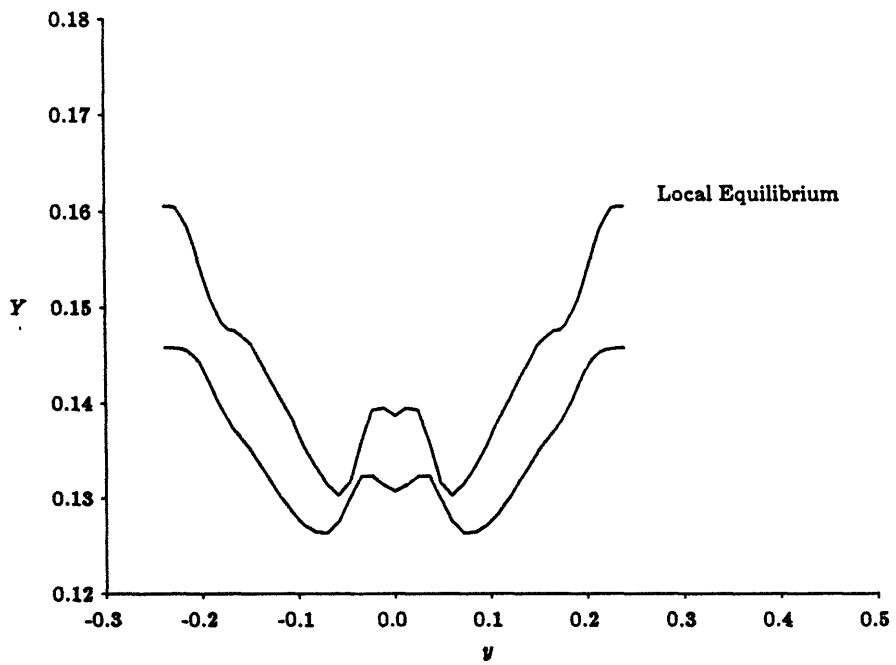

Figure 8.60: Distribution of steam mass fraction in the exit plane for two-strut scramjet inlet, premixed reacting flow.

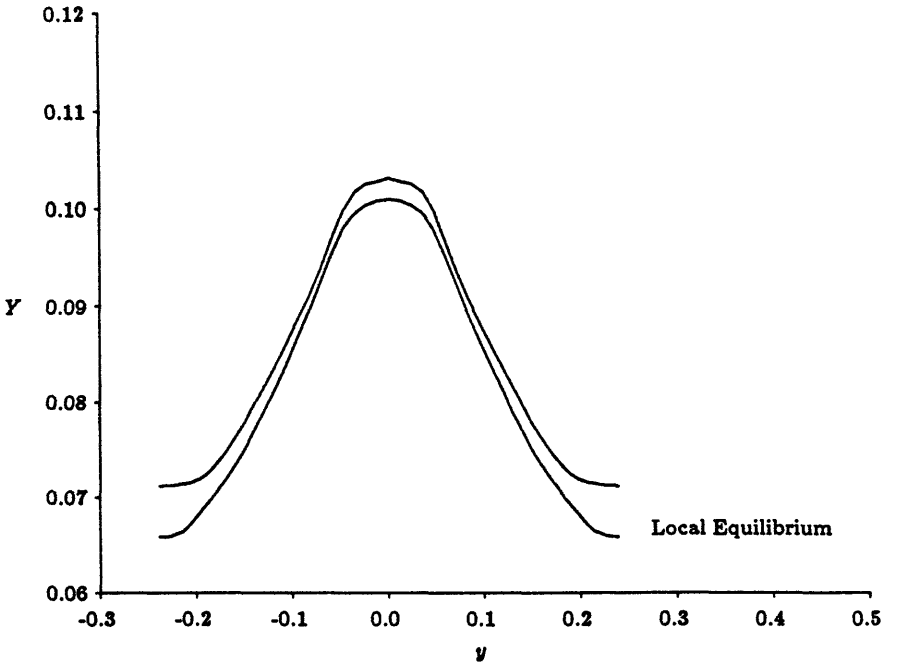

Figure 8.61: Distribution of mass fraction of oxygen in the exit plane for two-strut scramjet inlet, premixed reacting flow.

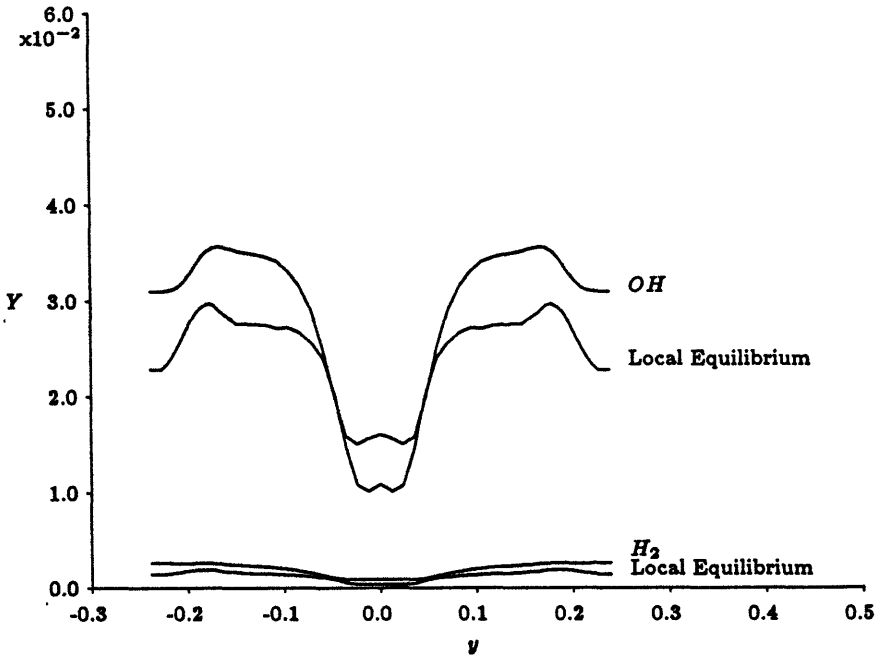

Figure 8.62: Distribution of mass fractions of hydroxyl and hydrogen in the exit plane for two-strut scramjet inlet, premixed reacting flow.

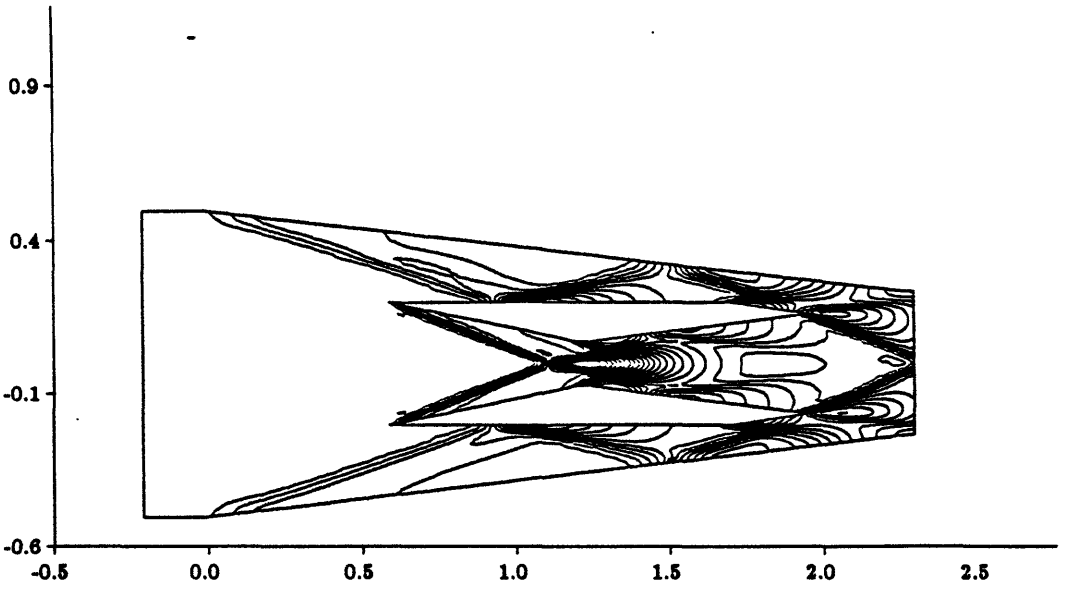

Figure 8.63: Density contours for two-strut scramjet inlet, premixed reacting flow.

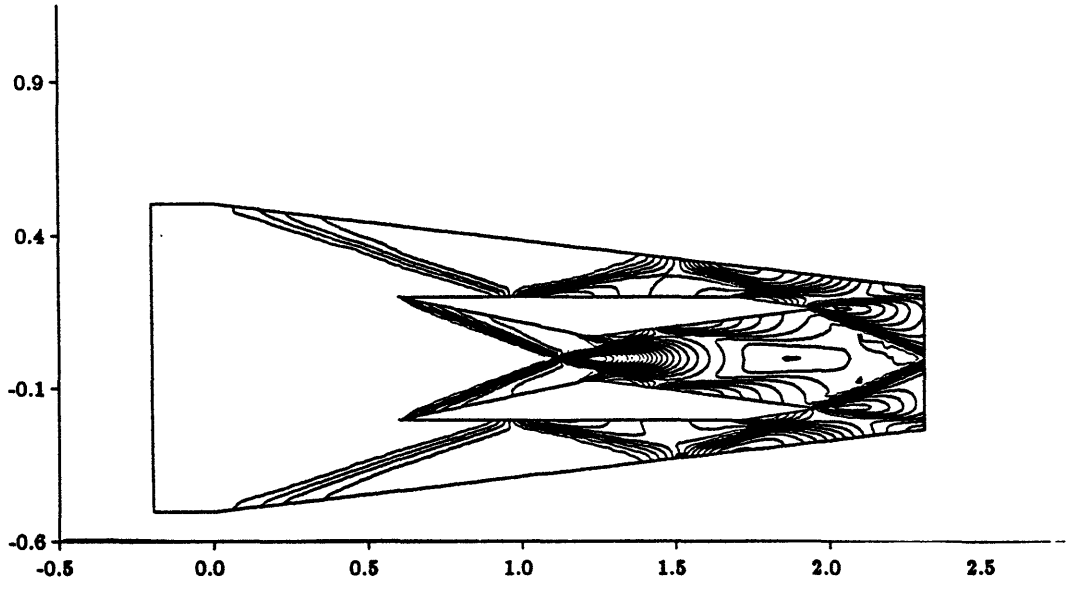

Figure 8.64: Pressure contours for two-strut scramjet inlet, premixed reacting flow.

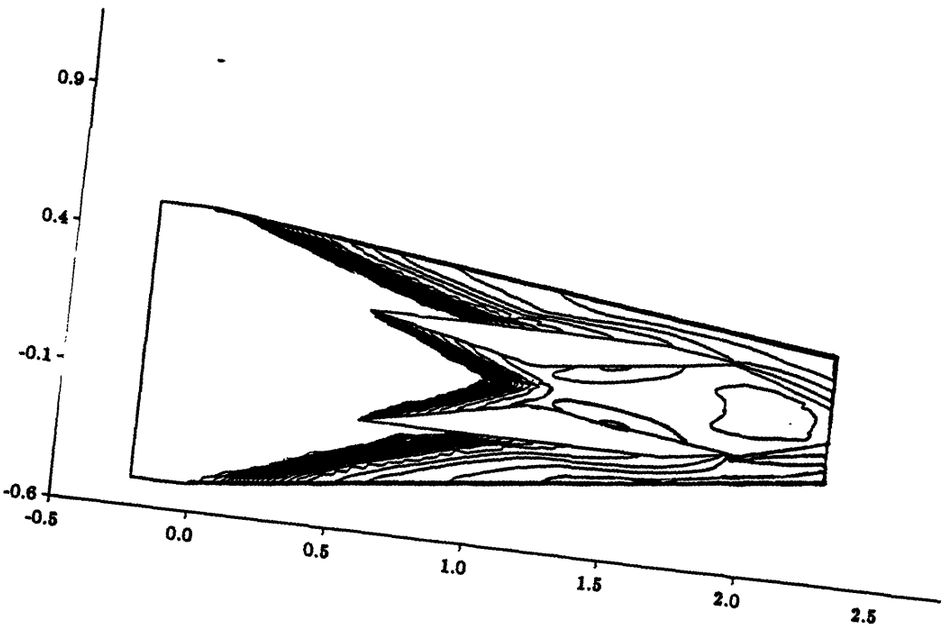

Figure 8.65: Oxygen mass fraction contours for two-strut scramjet inlet, premixed reacting flow.

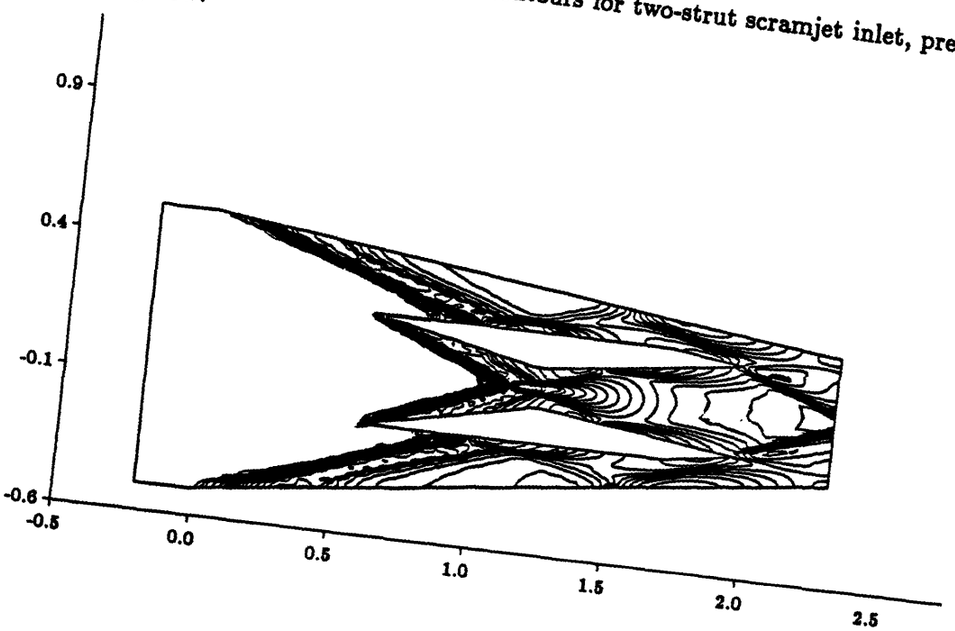

Figure 8.66: Hydroxyl mass fraction contours for two-strut scramjet inlet, premixed reacting flow.

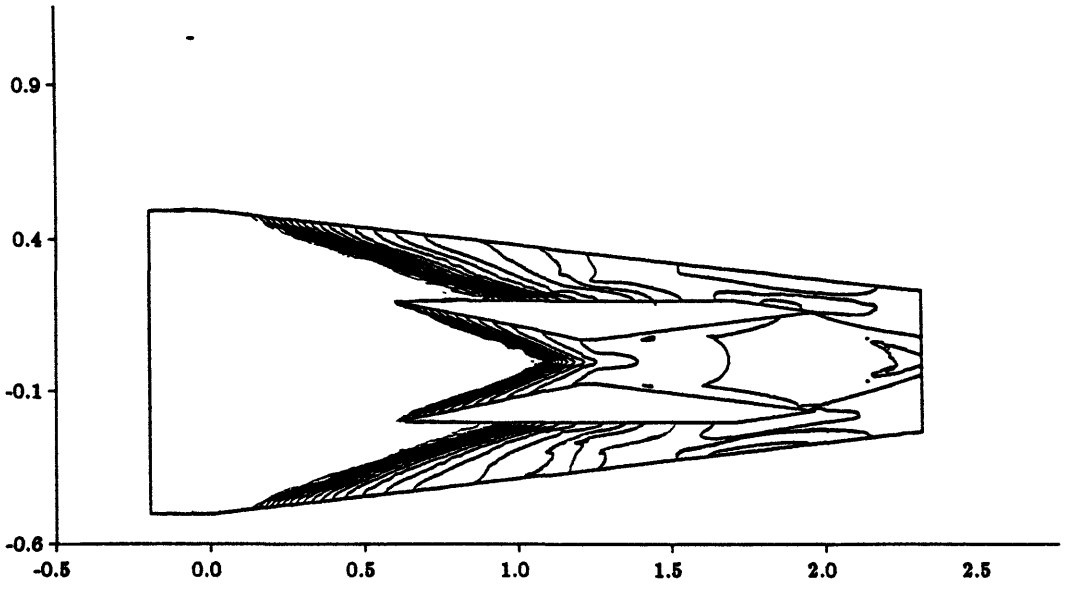

Figure 8.67: Hydrogen mass fraction contours for two-strut scramjet inlet, premixed reacting flow.

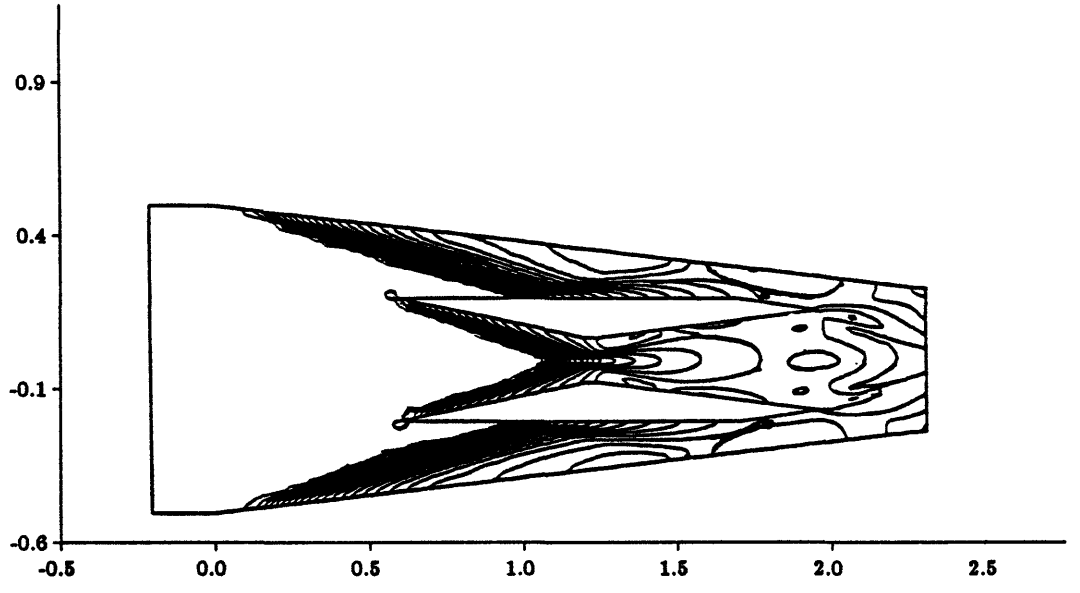

Figure 8.68: Steam mass fraction contours for two-strut scramjet inlet, premixed reacting flow.

the reacting case whereas the velocity and density were only slightly different. This additional pressure is due to the combustion process itself and it would be responsible in providing thrust to the vehicle. The variations of species mass fractions at the plane are shown in Figure (8.60) through (8.62). Also shown are the corresponding local equilibrium conditions. The contours of density, pressure and mass fractions are shown in Figure (8.63) through (8.68). The density and pressure contours indicate that the shocks are stronger for the reacting case. For example, for frozen flow the shock cross-over in between the two struts takes place at about $x = 1.2$ whereas that for the reacting case occurs near $x = 1.1$. The slip lines emanating from the trailing edges of the struts bend more towards the centerline than those in the frozen case. These figures also indicate that the reactions are much more pronounced immediately after the flow passes through the frontal shocks. However, the reactions do not go to completion in the computational domain. The average mass fraction of steam at the exit plane is about 0.135 compared to the maximum possible value of 0.205 for stoichiometric combustion and an equilibrium value of 0.143.

It is expected that additional combustion and expansion would take place in the "nozzle" part of the scramjet and would provide additional thrust. Although there is less than 0.3% hydrogen leaving the computational domain, there is some hydroxyl (average value 3%) and ample oxygen that can react to form steam in the nozzle part. Since the formation of steam is accompanied by heat release, additional thrust due to heat release could be expected.

8.3.3 Oscillating Inflow Example

In order to demonstrate the effectiveness of the spatio-temporal algorithm for multiple reactions, inflow conditions for a computational domain were varied sinusoidally. For this purpose a simpler geometry, as shown in Figure (8.69), was chosen and it represents a geometry similar to the central portion of the previous domain. The geometry is spanned by $x \in [-0.2, 1.4]$ and $y \in [0, 0.18]$ and the angle of the wedge is 14 degrees. Before allowing the inflow to vary temporally, a steady state flow was established for

which the inflow Mach number was assumed to be 4.308 with a temperature of 880 K and a pressure of 0.8 atmosphere. The reference length (distance between leading and trailing edges) was taken to be one meter. A total of three spatial levels were used for this case alongwith local time-stepping. Figure (8.69) shows the final grid for this case. Density and mass fraction of OH were used as the refinement parameters for spatial adaptation. The contours of density, pressure, temperature, x-component of velocity, local frozen Mach number, and the mass fractions of oxygen, hydroxyl, hydrogen, steam for the steady case are shown in Figure (8.70). The reactions start occurring after the first shock and relaxation regions can be seen clearly following this shock and its reflection from the symmetry axis. The production of steam is much pronounced after the second shock and its concentration remains relatively constant thereafter. It is observed that the species mass fractions remain nearly constant through the trailing edge expansion fan.

A periodic fluctuation was imposed on the mass flow at the inlet

$$\rho u = (\rho u)_0 \left[1 + A_m \sin(2\pi\omega t)\right] \tag{8.15}$$

where ω is the frequency and A_m is the amplitude of the oscillations; the subscript 0 indicates the value at time $t = 0$. Density, vertical velocity component and the energy term were fixed at the previous steady state values. For the numerical example these values were chosen to be as follows:

$$A_m = 0.1\,, \qquad \omega = 10 \tag{8.16}$$

Since one cycle corresponds to $\omega t = 1$, the time-period of the oscillations is 0.1. The solution was carried out until $t = 0.3$. Figure (8.71) shows the velocity variations on the upper channel wall at the end of each period whereas Figure (8.72) shows the variations of the mass fraction of steam on this wall. Computations were not carried out beyond $x = 1.0$ to save CPU time. It is observed that the oscillations have increased the combustion level to some extent (by about 3%). For these computations five levels of temporal embedding were used. These figures indicate that a *quasi-steady state* or a *periodic* solution has not yet evolved; however, the flow till about $x = 0.4$ seems to exhibit periodic behavior. Further note that as the velocity of the flow field reduces, the

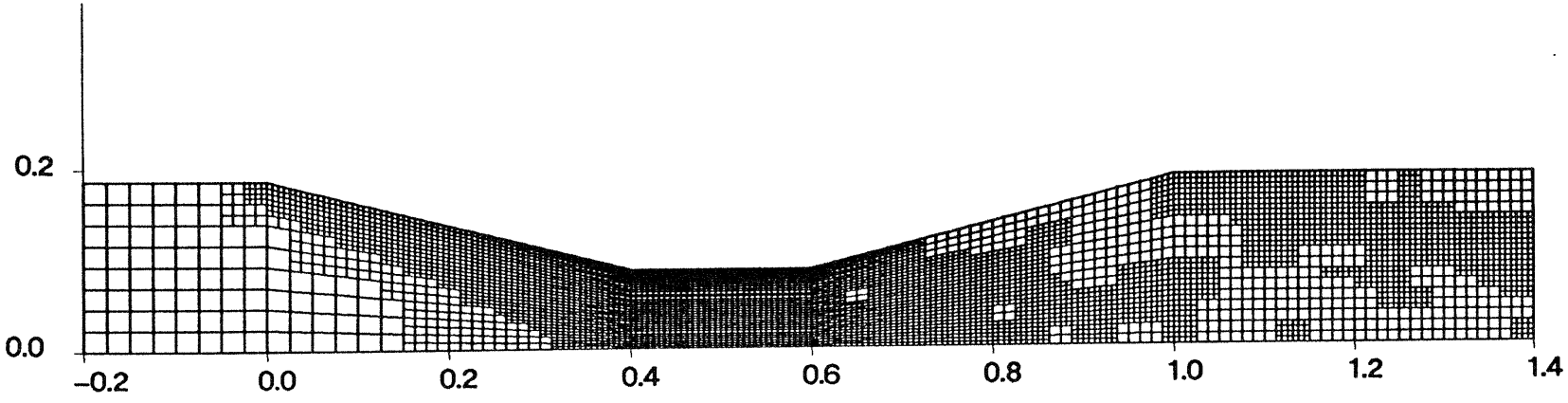

Figure 8.69: Final grid for steady state solution in an inlet, $M_i = 4.308$, premixed reacting flow.

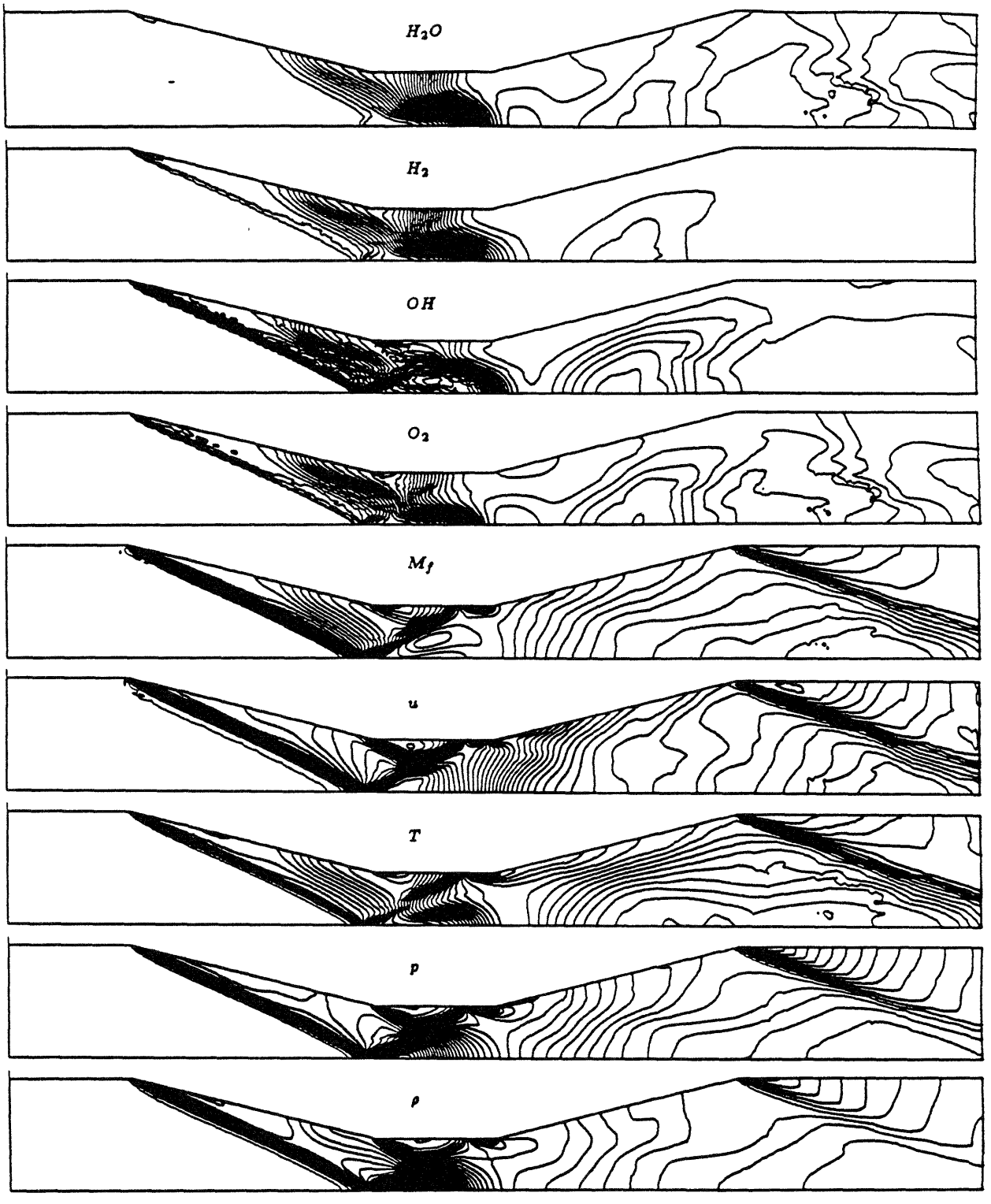

Figure 8.70: Contours for flow variables through an inlet for steady state solution, $M_i = 4.308$, premixed reacting flow.

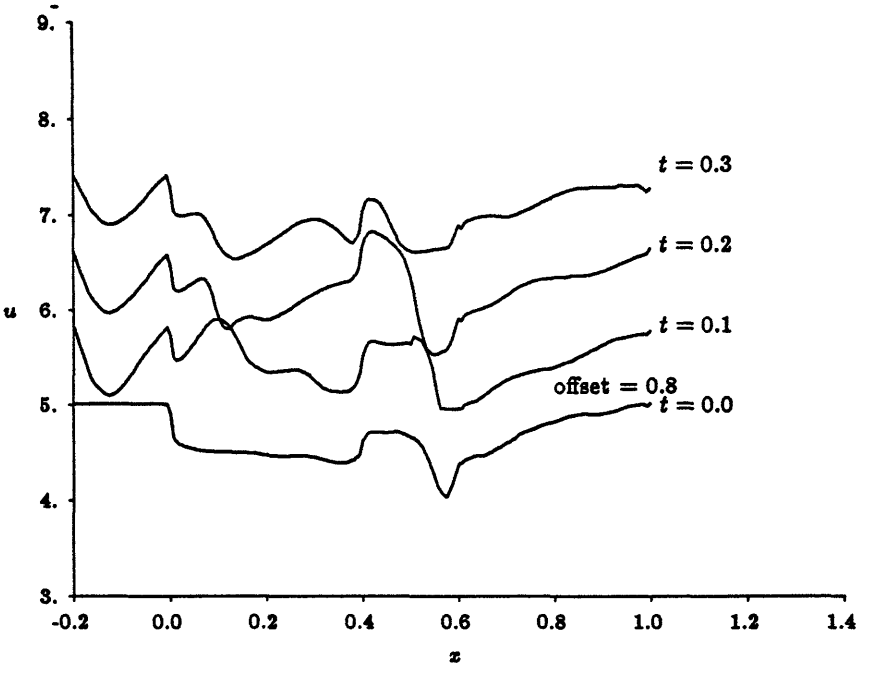

Figure 8.71: Velocity variations at upper channel wall for oscillating flow.

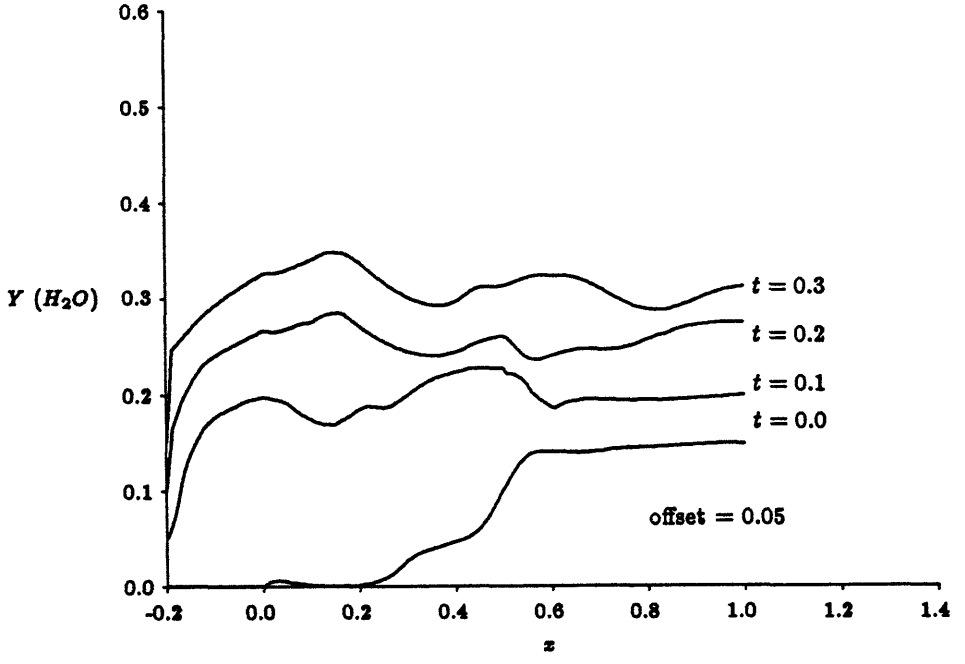

Figure 8.72: Steam mass fraction variations at upper channel wall for oscillating flow.

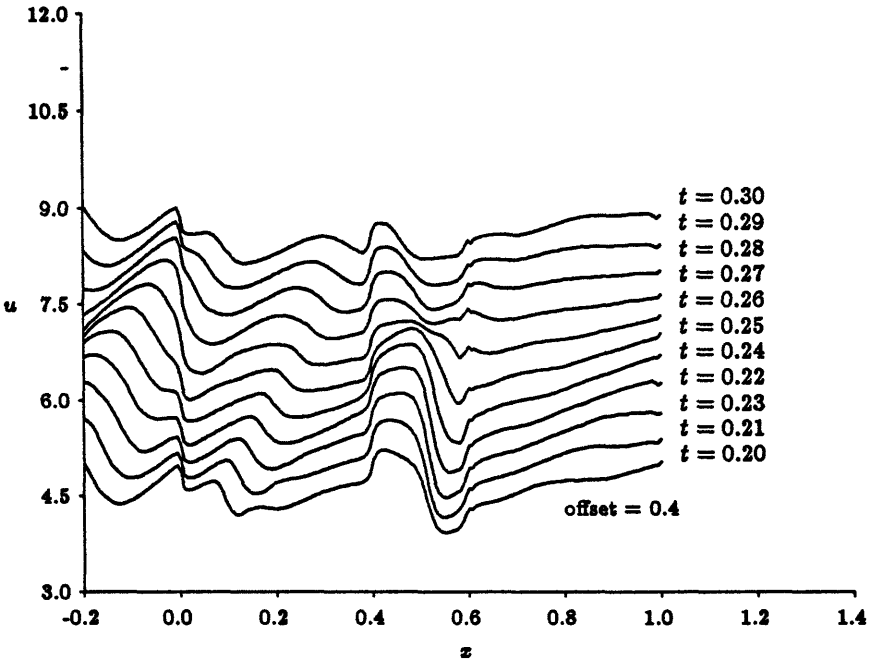

Figure 8.73: Velocity variations at upper channel wall for oscillating flow, for $t = 0.2, 0.3$.

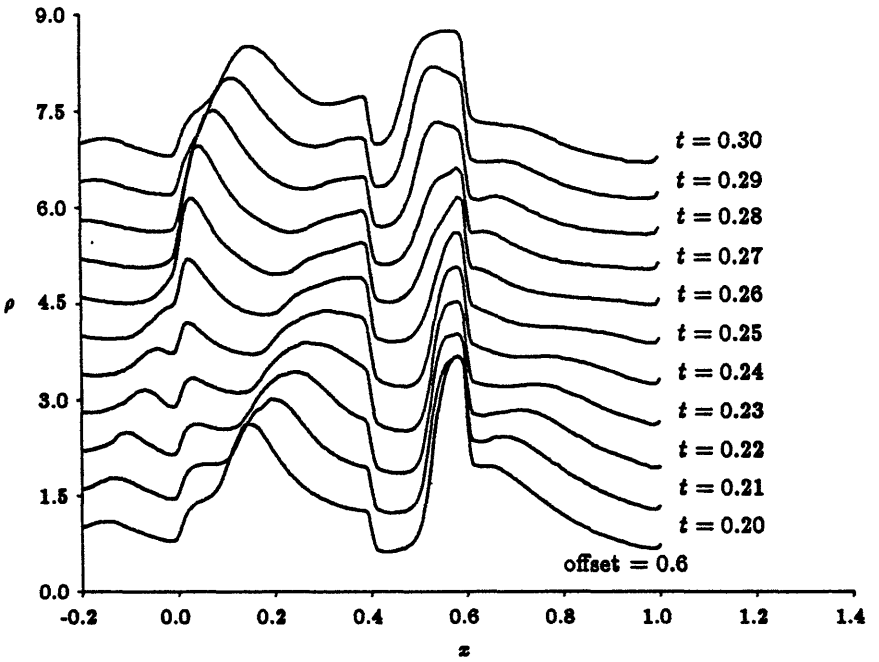

Figure 8.74: Density variations at upper channel wall for oscillating flow, for $t = 0.2, 0.3$.

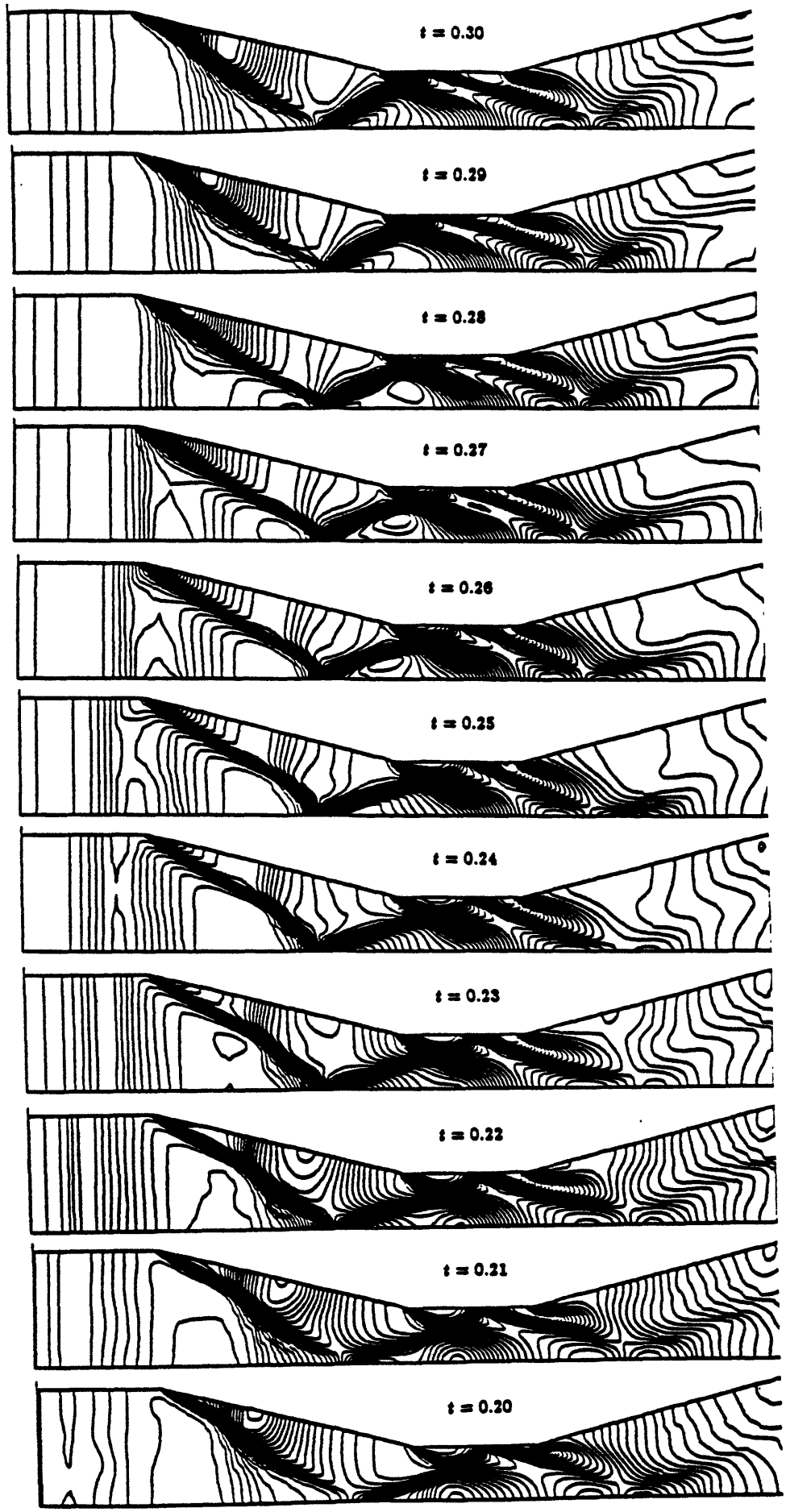

Figure 8.75: Contours for density for oscillating flow between $t = 0.2$ and 0.3.

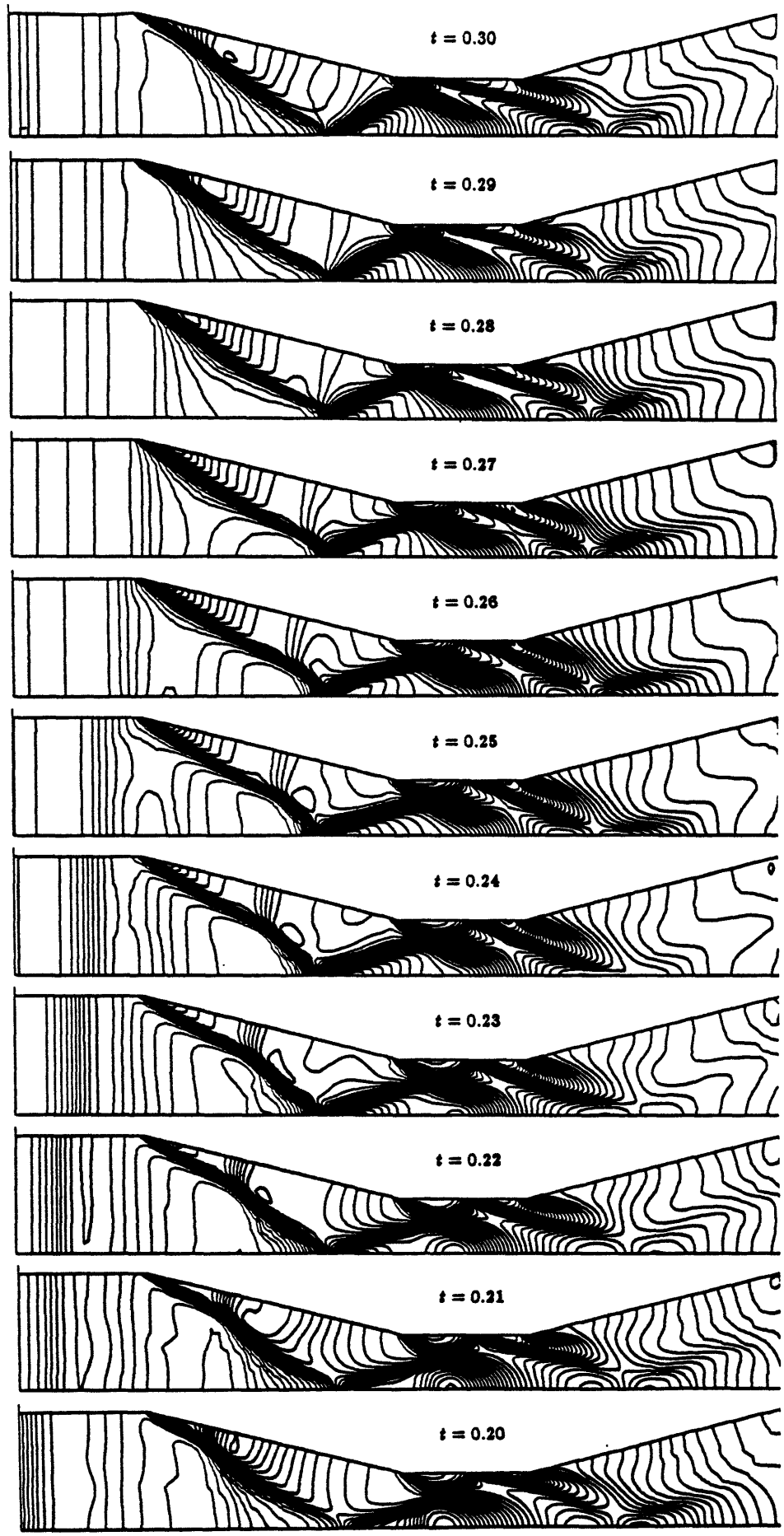

Figure 8.76: Contours for pressure for oscillating flow between $t = 0.2$ and 0.3.

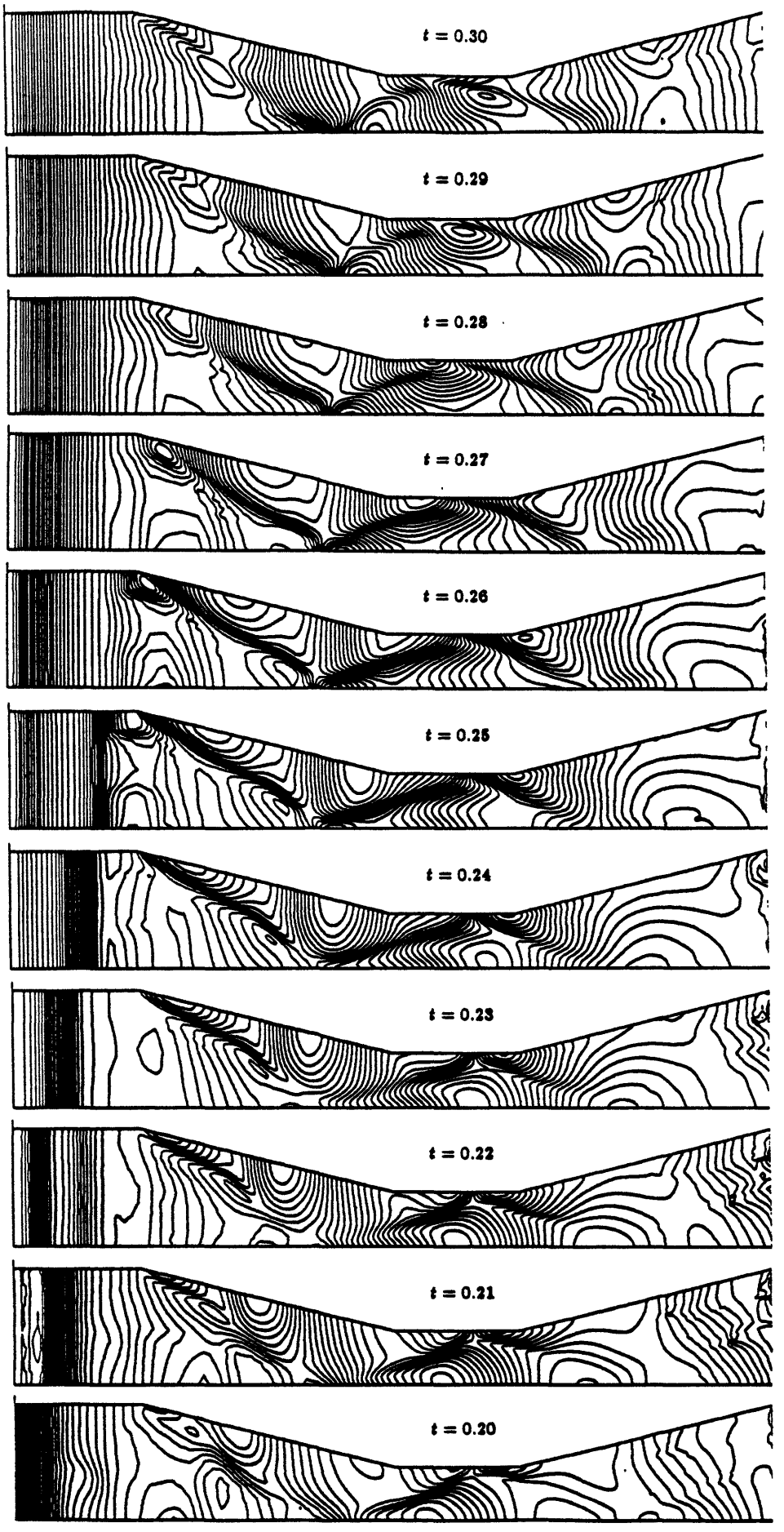

Figure 8.77: Contours for mass fraction of steam for oscillating flow between $t = 0.2$ and 0.3.

temperature increases and combustion starts occurring before the flow passes through the first shock. The velocity and density variations on the upper channel wall for the second cycle are shown in Figures (8.73) and (8.74). The contours of density, pressure and mass fraction of steam are shown in Figures (8.75) through (8.77). These contours indicate that the initially straight shock, emanating from the leading edge, changes as the disturbance passes across it. It is observed that the disturbance at the base of the corner affects the shock location at the symmetry axis and hence the reflected shocks are changed somewhat. Although the overall local frozen Mach number at the inlet plane varies from about 2.5 to 6.0, the flow-field exihits small changes with respect to the mean flow. This is due to the fact that the frequency of oscillations is high and the mean flow aft of the initial shock remains fairly stable. The contours of mass fractions of steam clearly indicate substantial changes through the reflected shocks at various positions and locations with small changes in density and pressure.

Chapter 9

Concluding Remarks

9.1 Summary

This thesis has examined predominantly supersonic reacting flows in which the transport effects have been neglected. A strategy has been developed for automatic spatial and temporal grid embedding for a reacting flow in both quasi-one-dimensional and two-dimensional situations. The unique part of the work, relative to previous studies, is the development of the temporal adaptation procedure and its coupling with spatial adaptation for unsteady chemically reacting or frozen flow systems. A new procedure for utilizing the first differences of more than one variable, to determine the allocation of spatial resolution, is also presented. Furthermore, a procedure for the selection of time-steps for source implicit schemes is detailed that switches the time-steps from small values when rapid temporal changes occur to large values when the temporal gradients diminish. Emphasis is placed on understanding supersonic combustion of hydrogen in air and moving blast waves in perfect or dissociating gases.

The algorithm periodically examines the evolving numerical solution, applies spatial adaptation to the existing grid, determines an appropriate time-stepping sequence for each cell in order to make up consistent time-stride units for the entire domain, and finally integrates the equations.

The spatial adaptation procedure consists of the following sequential operations:

1. local embedding or grid division,
2. extension of spatially embedded regions,

3. fusion of cells in other regions, and

4. removal of the knottiness in the grid by avoiding islands and voids.

Local embedding is carried out by detecting the regions of large spatial non-uniformities and subsequent subdivision of the corresponding grids. This spatial resolution is added over the entire domain prior to the execution of each temporal cycle, and is based upon first differences of the density and/or mass fractions of appropriate species. The procedure limits the cell volumes to four to one ratios for any set of contiguous cells. When the initial flow field on a coarse grid involves spatial non-uniformities, consistent pre-embedding is applied so as not to degrade this initial field.

Since the movement of flow features may be very large for certain unsteady applications, it is necessary to extend the spatially resolved region by a certain number of cells to ensure that the flow features will remain within this resolved region during the next time-stride unit. In general the larger the disparity of overall cell time-steps the more should be the number of layers of extension cells. The addition of buffer layers is accomplished by first determining the current set of the divided cells and then refining those coarse cells which are outside and adjacent to be identical in spatial resolution to those just inside the boundary, and repeating this process a specified number of times.

The procedure allows for both grid refinement and a return to the coarser mesh, within some specified coarsest global spatial grid. It is important for unsteady flows to allow for a cell fusion capability since otherwise grids might become uniformly fine after a while and the advantages of dynamic embedding would be lost. The coarsening of cells is also accomplished by examining the first differences of density and/or mass fractions. When these differences diminish on a previously refined grid, and become less than a critical limit, those contiguous grids which had been previously generated from the same parent cells may be fused.

After the alterations are completed in the spatial grid structures, a sequence of time-steps is determined for all the cells in the domain. The cells with the same time-step are integrated and updated together on different integration passes of the temporal

adaptation cycle. Once all the integration passes are completed, all the nodes in the domain arrive at the same time value and a time-stride is completed.

As part of the determination of the cell time-steps, the temporal gradients are monitored so as to maintain sufficiently small time-steps for adequate local resolution and stability. The time-step resolution takes into account the classical CFL restriction and the requirement implied by constraining the anticipated cell change to a small value. The temporal adaptation procedure allows for a maximum factor of four in local time-steps between contiguous cells. The overall disparity of the cell time-steps could be much higher.

To maintain time accuracy the total number of integrations for cells with smaller time-steps is carried out more often compared to those with larger time-steps. The cells are divided into subsets as characterized by their time-steps. The cells within each subset are integrated and updated together and a sequence of integration for cells in these subsets avoids integrating the same cells consecutively.

When the reactive equations are stiff in the sense that numerical stability rather than accuracy dictates the time-steps, then an implicit scheme can be used to partially alleviate the computational overheads. The time resolution criterion as proposed in this thesis limits the time-steps to small values during the earlier periods of a relaxation process when the temporal changes are large. The initial cell changes may be large due to the fact that the departure from local equilibrium conditions is large for some fast reactions and that the flux terms are not in balance with the source terms. However, as time elapses, the temporal gradients degrade, due to a new balance between the source and flux terms, although the departure from equilibrium could still be significant. For these relaxing cases larger time-steps, compared to those dictated by an explicit stability criterion based upon chemical source terms, can be used to advance the solution by utilizing an implicit scheme. The same implicit scheme can also be used when the time-steps have to be reduced to capture rapid relaxation phenomenon.

Depending upon the rate of variations of the flow features, the spatial adaptation may follow after the temporal adjustment or a number of time-strides may be carried

out prior to the next spatial adjustment of the grids. The number of time-strides between two consecutive spatial adaptation procedures is user-controlled rather than being dynamically computed by the algorithm, since it is highly problem dependent. The user is generally aware of an expected rate of variations of feature properties and s/he could request the spatial and temporal procedures to alternate each other in a limiting scenario. The integration of the equations continue until a desired number of time-strides is completed or when the time-level exceeds some user-supplied value.

9.2 Conclusions and Discussion

Adaptive embedding algorithms have the advantage that meshes are refined only where necessary and as the solution evolves, thereby providing accurate and relatively inexpensive solutions. Since the local embedding can be carried out in a recursive manner, very fine grid spacing can be maintained in the vicinity of the physical structures being captured. Furthermore, since the resolution is enhanced only locally at the features, with coarser grids near successively uniform flow regions, the computations with such grids consume significantly less computer resources than does global refinement. There are substantial savings in both CPU time and memory.

Just as different spatial resolutions are allocated at different locations of a spatial grid to achieve CPU time gains, it is beneficial to take advantage of the large spatial variations of time-steps for frozen or reacting flows. In fact gains due to utilization of different time-steps can even be achieved for unsteady frozen flows if there exist substantial variations in spatial cell volumes, which indeed may well be a result of spatial adaptation. It is clear from the CFL constraint that the resolution requirements in space generally imply a corresponding imposition on resolution in time. For most frozen flows this is the primary constraint, but for reacting flows other temporal resolution requirements may be even more stringent than those implied by the spatial resolution. Similarly for strong blast waves the maximum eigenvalues can change by an order of magnitude across a shock and for these cases the temporal adaptation could be beneficial even for frozen flows on uniform grids. In general, the larger the global disparity of

the cell time-steps the more effective is the temporal adaptation, as is true for spatial adaptation.

In chemically reacting flows, the computations of chemical kinetic terms is often more expensive than evaluations of convective and/or diffusive transport terms. The cost increases with the number of species, the number of reactions connecting these species, the number of spatial cells and the inverse of the time-step size. For flame and detonation simulations the overall calculation may take two or more orders of magnitude longer compared to frozen flow situations. Calculations may also be costly due to stiffness introduced into the equations by the finite rate chemical kinetics which may be necessary to describe the physical situation. The utilization adaptive grids in both space and time for such flows can lead to orders of magnitude savings in the CPU time.

Separate pointer systems for both spatial and temporal adaptation procedures and chemistry manipulations are utilized for the current algorithm. The spatial data base tallies the spatial level, supercell, and the surrounding nodes of each cell in the domain. Similarly, information about cells adjacent to each node must be known and boundary points must carry details like boundary condition type, adjacent node and cells, *etc.* The temporal data base tracks the number of cells and the sequence of integration during each time-stride. This pointer system must be updated after each time-stride for assignments of time-steps, determination of the temporal level of cells and their allocation into clusters classified by these levels, determination of nodits, and constraining of time-steps among contiguous cells to four to one ratios. Some of this represents an overhead but when compared to the gain achieved in efficiency proves to be well worth doing. The chemistry data structure holds information for each species in the model, for example, specific heat, heat of formation, *etc.* and information pertaining to each reaction, for example, constants in Arhenius rate model, total number of species, *etc.* The data structure also keeps track of the table of species involved in specific reactions and all the stoichiometric coefficients.

Depending upon the problem, the spatial data base updating may not be required as

frequently as that for the temporal data base. For steady state problems the number of changes in the spatial pointer system generally equals the number of spatially embedded levels desired and the adaptation can be performed at either specified iteration intervals or residual levels. Similarly, for unsteady problems in which the characteristic feature speeds are relatively small the adjustments to the spatial pointer system are infrequent. However, when high feature speeds arise, either the time-stride size must be kept small or the spatially embedded clusters enlarged, so that the features do not move out of their respective clusters during a given time-stride. The process of enlarging spatially embedded clusters can become computationally expensive; a balance is required between these competing effects. For unsteady flows, spatial adaptation procedure must be applied frequently because the features to be resolved may be moving and the adaptive grid clearly must track these features at a synchronous speed.

For all of the sample cases the numerical solutions based on an adaptation procedure were comparable in accuracy to globally fine grid solutions, and were in good agreement with previous works. Computed examples also indicate that the numerical solution obtained by utilizing spatio-temporal algorithm can yield orders of magnitude faster computations compared to those of globally fine grids. The CPU time savings increase with the increase in the number of spatial and/or temporal levels of embedding. For unsteady flow exmaples the adaptive grid clearly tracks the salient features at a synchronous speed and is capable of resolving features like shocks, relaxation zones, slip lines, *etc.*

9.3 Future Extensions

Since the savings in CPU time increase substantially from quasi-one-dimensional to two-dimensional studies, it does appear promising to introduce temporal adaptation concurrently with spatial adaptation for three-dimensional, unsteady reacting flow fields. There appears to be little theoretical difficulty in extending the present adaptive grid algorithm to a third spatial dimension. However, this might only be practical for moderate sized problems to run on a machine in the supercomputer class.

While the present work is concerned with the solution of Euler equations, Chima and Johnson [29] and Davis [39] have demonstrated that Ni's scheme is extendible to the Reynold's averaged transport equations. Furthermore, Kallinderis and Baron [71] have developed Ni scheme to include transport effects and an adaptive procedure when interest is limited to steady state problems. The spatio-temporal algorithm developed here should prove to be an attractive option for calculations involving embedded viscous regions.

Appendix A

Jacobians, Eigenvalues, Eigenvectors

A.1 Analytic Jacobians of Flux Vectors

The Jacobian matrices F_U, G_U, W_U are required for the integration of the partial differential equations. For the purpose of evaluating the flux Jacobians, a calorically perfect gas mixture will be assumed, *i.e.*, the specific heat of each species in the mixture will be regarded constant. Once the Jacobian terms are derived, local frozen values can be substituted in place of constant values. The Jacobian evaluations will be shown here only for the two-dimensional case.

The notation used in this section is as follows. The components of the vectors U, F, G, W are indicated by numbered subscripts. For example, $U_1 = \rho$, $F_2 = \rho u^2 + p$, *etc.* Double subscripts indicate the Jacobian elements, *e.g.*, $F_{21} = \frac{\partial F_2}{\partial U_1}$. The pressure term p_i stands for $\frac{\partial p}{\partial U_i}$. The total number of equations to be solved is denoted by N_e, so the species equations correspond to the components $k = 4+s$ where $1 \leq s \leq N_e - 4 \leq S-1$. In what follows the elements of the flux vectors will be written in terms of both primitive variables and components of the state vector.

$$F_1 = \rho u = U_2$$

$$F_{1j} = \begin{cases} 1 & j = 2 \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

$$F_2 = \rho u^2 + p = \frac{U_2^2}{U_1} + p$$

$$F_{2j} = \begin{cases} -u^2 + p_1 & j = 1 \\ 2u + p_2 & j = 2 \\ p_j & \text{otherwise} \end{cases} \tag{A.2}$$

The partial derivatives of pressure will be determined latter.

$$F_3 = \rho uv = \frac{U_2 U_3}{U_1}$$

$$F_{3j} = \begin{cases} -uv & j = 1 \\ v & j = 2 \\ u & j = 3 \\ 0 & \text{otherwise} \end{cases} \tag{A.3}$$

$$F_4 = (p+\epsilon)u = \frac{U_2 F_2}{U_1} + \frac{U_2 U_4}{U_1} - \frac{U_2^3}{U_1^2}$$

$$F_{4j} = \begin{cases} u\left(F_{21} + u^2 - \frac{p+\epsilon}{\rho}\right) & j = 1 \\ uF_{22} - 2u^2 + \frac{p+\epsilon}{\rho} & j = 2 \\ u\left(F_{24} + 1\right) & j = 4 \\ uF_{2j} & \text{otherwise} \end{cases} \tag{A.4}$$

$$F_k = F_{4+s} = \rho u Y_s = \frac{U_2 U_k}{U_1}, \quad k = 5, \ldots, N_e$$

$$F_{kj} = \begin{cases} -uY_s & j = 1 \\ Y_s & j = 2 \\ u & j = k = 4 + s \\ 0 & \text{otherwise} \end{cases} \tag{A.5}$$

Note that the assumption of constant specific heats is not utilized until now; however, the assumption simplifies the partial derivatives of pressure, and the caloric equation of state (Eq. 2.55) then becomes

$$\epsilon = \sum_{s=1}^{S} \rho Y_s H_{f_s} + \frac{\rho}{2}(u^2 + v^2) + \rho T \sum_{s=1}^{S} Y_s C_{p_s} - \rho T_0 \sum_{s=1}^{S} Y_s C_{p_s} - p \tag{A.6}$$

On substituting the thermal equation of state for temperature this yields

$$p = \sum_{s=1}^{S} \rho Y_s H_{f_s} + \frac{\rho}{2}(u^2+v^2) + \frac{p}{\mathcal{R}\sum_s Y_s/\hat{m}_s} \sum_{s=1}^{S} Y_s C_{p_s} - \rho T_0 \sum_{s=1}^{S} Y_s C_{p_s} - \epsilon \qquad \text{(A.7)}$$

Since the specific heats are related by

$$C_{v_s} = C_{p_s} - \frac{\mathcal{R}}{\hat{m}_s} \qquad \text{(A.8)}$$

the pressure equation simplies to

$$p\frac{\sum_s Y_s C_{v_s}}{\mathcal{R}\sum_s Y_s/\hat{m}_s} = \epsilon - \sum_{s=1}^{S} \rho Y_s H_{f_s} - \frac{\rho}{2}(u^2+v^2) + \rho T_0 \sum_{s=1}^{S} Y_s C_{p_s} \qquad \text{(A.9)}$$

or in terms of the components of state vector

$$p\frac{\sum_s U_k C_{v_s}}{\mathcal{R}\sum_s U_k/\hat{m}_s} = U_4 - \sum_{s=1}^{S} U_k H_{f_s} - \frac{U_2^2+U_3^2}{2U_1} + T_0 \sum_{s=1}^{S} U_k C_{p_s} \ , \qquad k = 4+s \quad \text{(A.10)}$$

For derivatives with respect to U_1, U_2, U_3, U_4 all terms involving U_k are constants; hence the following mixture values can be defined

$$\begin{aligned} C_p &= \textstyle\sum_s Y_s C_{p_s} \ , & C_v &= \textstyle\sum_s Y_s C_{v_s} \\ \hat{m} &= \frac{1}{\sum_s Y_s/\hat{m}_s} \ , & H_f &= \textstyle\sum_s Y_s H_{f_s} \end{aligned} \qquad \text{(A.11)}$$

Hence the pressure equation becomes

$$p = \frac{\mathcal{R}}{\hat{m}C_v}\left\{U_4 - \frac{U_2^2+U_3^2}{2U_1}\right\} + K \qquad \text{(A.12)}$$

where K is a constant insofar as the first four derivatives are concerned and is given by

$$K = \frac{\mathcal{R}}{\hat{m}C_v}\{\rho T_0 C_p - \rho H_f\} \qquad \text{(A.13)}$$

Using the ratio of specific heats for a mixture, *i.e.*, $\gamma = \sum Y_s C_{p_s} / \sum Y_s C_{v_s}$, it follows that

$$\frac{\mathcal{R}}{\hat{m}C_v} = \frac{C_p - C_v}{C_v} = \gamma - 1 \qquad \text{(A.14)}$$

The first four partial derivatives of pressure then become

$$p_j = \begin{cases} \frac{V^2}{2}(\gamma-1) & j=1 \\ u(1-\gamma) & j=2 \\ v(1-\gamma) & j=3 \\ (\gamma-1) & j=4 \end{cases} \qquad \text{(A.15)}$$

Thus the first four Jacobians of F_2 now become

$$F_{2j} = \begin{cases} \frac{\gamma-3}{2}u^2 + \frac{\gamma-1}{2}v^2 & j=1 \\ (3-\gamma)u & j=2 \\ (1-\gamma)v & j=3 \\ \gamma - 1 & j=4 \end{cases} \tag{A.16}$$

Similarly the first four Jacobians of F_4 become

$$F_{4j} = \begin{cases} u\left(\frac{\gamma-1}{2}V^2 - \frac{p+\epsilon}{\rho}\right) & j=1 \\ (1-\gamma)u^2 + \frac{p+\epsilon}{\rho} & j=2 \\ (1-\gamma)uv & j=3 \\ u\gamma & j=4 \end{cases} \tag{A.17}$$

For derivative of p with respect to U_l, with $l = 4+q \in [5, N_e]$, the quantities K, C_p, C_v are not constants. Thus from Equation (A.10)

$$\frac{p_l}{\gamma-1} + \frac{p}{\left(\sum_s \frac{RU_k}{\hat{m}_s}\right)^2}\left\{\sum_s \left(\frac{RU_k}{\hat{m}_s}\right) C_{v_q} - \sum_s (U_k C_{v_s})\frac{\mathcal{R}}{\hat{m}_q}\right\} = -H_{f_q} + T_0 C_{p_q} \tag{A.18}$$

which can be simplified to

$$p_l = (\gamma-1)(T_0 C_{p_q} - H_{f_q}) + \frac{p}{\rho}\frac{\hat{m}}{\hat{m}_q}\left(\frac{\gamma_q - \gamma}{\gamma_q - 1}\right) \tag{A.19}$$

Replacing q by s gives the following

$$F_{2(4+s)} = (\gamma-1)(T_0 C_{p_s} - H_{f_s}) + \frac{p}{\rho}\frac{\hat{m}}{\hat{m}_s}\left(\frac{\gamma_s - \gamma}{\gamma_s - 1}\right) \tag{A.20}$$

This completes the expressions for the Jacobians of the flux vector F. In summary the

matrix F_U can be written as

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\frac{\gamma-1}{2}V^2 - u^2 & (3-\gamma)u & (1-\gamma)v & \gamma-1 & F_{25} & F_{26} & F_{27} & \cdots \\
-uv & v & u & 0 & 0 & 0 & 0 & \cdots \\
u\left(\frac{\gamma-1}{2}V^2 - \frac{p+\epsilon}{\rho}\right) & \frac{p+\epsilon}{\rho} + (1-\gamma)u^2 & uv(1-\gamma) & u\gamma & uF_{25} & uF_{26} & uF_{27} & \cdots \\
-uY_1 & Y_1 & 0 & 0 & u & 0 & 0 & \cdots \\
-uY_2 & Y_2 & 0 & 0 & 0 & u & 0 & \cdots \\
-uY_3 & Y_3 & 0 & 0 & 0 & 0 & u & \cdots \\
 & \cdots & & \cdots & & & & \cdots
\end{bmatrix} \tag{A.21}$$

The evaluation of the Jacobians of flux vector G will now be described.

$$G_1 = \rho v = U_3$$

$$G_{1j} = \begin{cases} 1 & j = 3 \\ 0 & \text{otherwise} \end{cases} \tag{A.22}$$

$$G_2 = F_3 = \rho uv = \frac{U_2 U_3}{U_1}$$

$$G_{2j} = F_{3j} \qquad j = 1, \ldots, N_e \tag{A.23}$$

$$G_3 = \rho v^2 + p = F_2 + \frac{U_3^2}{U_1} - \frac{U_2^2}{U_1}$$

$$G_{3j} = \begin{cases} F_{21} - v^2 + u^2 = \frac{\gamma-3}{2}v^2 + \frac{\gamma-1}{2}u^2 & j = 1 \\ F_{22} - 2u = (1-\gamma)u & j = 2 \\ F_{23} + 2v = (3-\gamma)v & j = 3 \\ F_{24} = \gamma - 1 & j = 4 \\ F_{2j} & \text{otherwise} \end{cases} \tag{A.24}$$

$$G_4 = \quad (p+\epsilon)v = \frac{U_3 F_2}{U_1} + \frac{U_3 U_4}{U_1} - \frac{U_2^2 U_3}{U_1^2}$$

$$G_{4j} = \begin{cases} v(F_{21} + u^2 - \frac{p+\epsilon}{\rho}) = v\left(\frac{\gamma-1}{2}V^2 - \frac{p+\epsilon}{\rho}\right) & j = 1 \\ v(F_{22} - 2u) = (1-\gamma)uv & j = 2 \\ vF_{23} + \frac{p+\epsilon}{\rho} = (1-\gamma)v^2 + \frac{p+\epsilon}{\rho} & j = 3 \\ v(F_{24} + 1) = v\gamma & j = 4 \\ vF_{2j} & \text{otherwise} \end{cases} \tag{A.25}$$

$$G_k = \quad G_{4+s} = \rho v Y_s = \frac{U_3 U_k}{U_1} \qquad k = 5, \ldots, N_e$$

$$G_{kj} = \begin{cases} -vY_s & j = 1 \\ Y_s & j = 3 \\ v & j = k = 4+s \\ 0 & \text{otherwise} \end{cases} \tag{A.26}$$

The Jacobians of the source vector W change from one reaction system to another. For the sake of generality the source vector Jacobians are evaluated numerically from the discrete form

$$\frac{\partial W_i}{\partial U_j} = \frac{W_i(U_1, \ldots, U_j + \Delta U_j, \ldots, U_{N_e}) - W_i(U_1, \ldots, U_j - \Delta U_j, \ldots, U_{N_e})}{2\Delta U_j} \tag{A.27}$$

where

$$\Delta U_j = \begin{cases} 0.001 U_j & , \quad U_j \neq 0 \\ 0.001 & , \quad \text{otherwise} \end{cases}$$

A.2 Eigenvalues of Jacobian Matrices

The eigenvalues of F_U and G_U are needed to determine the maximum allowable time-step and to apply the characteristic boundary conditions. The eigenvalues of F_U for the non-reacting case are $u + a, u - a, u, u$ where a is the frozen speed of sound. The two u eigenvalues are due to the continuity and y-momentum equations. Since

the species equations are essentially continuity equations, the total number of multiple roots for the reacting system is $S+1$ (1 for continuity, 1 for y-momentum and $S-1$ for species equations). Intuitively, the other two roots are expected to be $u \pm a_f$, where a_f is the local frozen speed of sound. One can expand and solve for the polynomial function corresponding to the eigenvalues of Equation (A.21), but it is simpler to evaluate the determinant of the matrix F_U as a product of the eigenvalues, *i.e.*,

$$|F_U| = (u^2 - c^2)u^{S+1} \tag{A.28}$$

where c is a speed which will be shown to be the local frozen speed of sound. To justify the assertion that the eigenvalues of F_U are really $u^{S+1}, u \pm c$, consider the trace of F_U, *i.e.*, the sum of the eigenvalues

$$(S+1)u + (u+c) + (u-c) \stackrel{?}{=} F_{22} + F_{33} + F_{44} + \sum_{s=1}^{S-1} u \tag{A.29}$$

This implies that

$$F_{22} + F_{33} + F_{44} \stackrel{?}{=} 4u \tag{A.30}$$

Substitution of F_{jj} values into this equation confirms the assertion.

The determinant of the Equation (A.21) can be shown to be

$$|F_U| = u^{S+1}\left\{(1-\gamma)\frac{p+\epsilon}{\rho} + \frac{\gamma+1}{2}u^2 + \frac{\gamma-1}{2}v^2 - \sum_s Y_s F_{2(4+s)}\right\} \tag{A.31}$$

From Equation (A.7) one can show that

$$\frac{p+\epsilon}{\rho} = \frac{\gamma}{\gamma-1}\frac{p}{\rho} + \frac{u^2+v^2}{2} + \sum_s Y_s H_{f_s} - T_0 \sum_s Y_s C_{p_s} \tag{A.32}$$

Substituting Equation (A.20) and (A.32) into (A.31) yields

$$|F_U| = u^{S+1}\left\{u^2 - \frac{\gamma p}{\rho} - \frac{p}{\rho}\hat{m}\sum_{s=1}^{S}\frac{Y_s}{\hat{m}_s}\left(\frac{\gamma_s - \gamma}{\gamma_s - 1}\right)\right\} \tag{A.33}$$

It can be verified that for an ideal mixture the last term inside the curly bracket vanishes. Note that

$$\begin{aligned}
\sum_{s=1}^{S}\frac{Y_s}{\hat{m}_s}\left(\frac{\gamma_s-\gamma}{\gamma_s-1}\right) &= \frac{1}{\hat{m}} - \sum\frac{Y_s}{\hat{m}_s}\left(\frac{\gamma-1}{\gamma_s-1}\right) = \frac{1}{\hat{m}} - (\gamma-1)\sum\frac{Y_s}{\hat{m}_s}\frac{C_{v_s}}{C_{p_s}-C_{v_s}} \\
&= \frac{1}{\hat{m}} - (\gamma-1)\sum\frac{Y_s C_{v_s}}{\mathcal{R}} = \frac{1}{\hat{m}} - \frac{\gamma-1}{\mathcal{R}}C_v = \frac{1}{\hat{m}} - \frac{C_p - C_v}{\mathcal{R}} \\
&= \frac{1}{\hat{m}} - \frac{1}{\hat{m}} = 0
\end{aligned} \tag{A.34}$$

Hence

$$|F_U| = u^{S+1}\left(u^2 - \frac{\gamma p}{\rho}\right) \tag{A.35}$$

But $a_f^2 = \gamma p/\rho$ is the square of the local frozen speed of sound. In a similar manner it can be verified that the eigenvalues of G_U are $v^{S+1}, v \pm a_f$, *i.e.*,

$$|G_U| = v^{S+1}\left(v^2 - \frac{\gamma p}{\rho}\right) \tag{A.36}$$

A.3 Eigenvectors of Jacobian Matrices

The eigenvectors of the Jacobian matrix F_U are needed for the computations involving characteristic boundary conditions. Only *left* eigenvectors will be considered here. The eigenvectors L_i are numbered according to the eigenvalues $\lambda_i = u - a_f, u + a_f, u, \ldots, u$. The equations in this section will be given for both a general case and an ideal mixture (constant specific heats). The left eigenvector L_1 for $\lambda_1 = u - a_f$ is given by

$$L_1\left(F_U - \lambda_1 \overline{\overline{I}}\right) = L_1 A_1 = 0 \tag{A.37}$$

where the notation $A_i = F_U - \lambda_i \overline{\overline{I}}$ is used for simplicity. The product with the fourth column of A_1 implies

$$L_{12}F_{24} + L_{14}(F_{44} + a_f - u) = 0$$

Since the eigenvectors of a distinct eigenvalue are unique up to a multiplicative constant the choice $L_{14} = 1$ is made. Hence

$$L_{12} = \frac{u - a_f - F_{44}}{F_{24}} = -\left(u + \frac{a_f}{F_{24}}\right) = -\left(u + \frac{a_f}{\gamma - 1}\right) \tag{A.38}$$

The product with the third column of A_1 implies

$$L_{12}F_{23} + L_{13}a_f + L_{14}uF_{23} = 0$$

or

$$L_{13} = \frac{F_{23}}{F_{24}} = -v \tag{A.39}$$

The column of A_1 pertaining to species s implies

$$L_{12}F_{2k} + uL_{14}F_{2k} + a_f L_{1k} = 0 \ , \qquad k = 4 + s$$

or

$$L_{1k} = \frac{F_{2k}}{F_{24}} = T_0 C_{p_s} - H_{f_s} + \frac{p\hat{m}}{\rho \hat{m}_s}\left(\frac{\gamma_s - \gamma}{(\gamma-1)(\gamma_s - 1)}\right) \tag{A.40}$$

It can be verified that the first and second columns of A_1 yield redundant values of L_{11}. The result for second column is

$$L_{11} + L_{12}(F_{22} + a_f - u) + L_{13}v + L_{14}F_{42} + \sum_{k=5}^{N_s} Y_s L_{1k} = 0 \ , \qquad s = k-4$$

This can be simplified to

$$L_{11} = V^2 + \frac{a_f}{\gamma - 1}(a_f + u) - \frac{p+\epsilon}{\rho} - \sum_{k=5}^{N_s} Y_s L_{1k} \ , \qquad s = k-4 \tag{A.41}$$

This can be further simplified by substituting the values of ϵ/ρ and L_{1k} and hence

$$L_{11} = \frac{u}{2}\left(u + \frac{2}{\gamma - 1}a_f\right) + \frac{v^2}{2} \tag{A.42}$$

The left eigenvector L_2 for $\lambda_1 = u + a_f$ is given by

$$L_2\left(F_U - \lambda_2 \bar{\bar{I}}\right) = L_2 A_2 = 0 \tag{A.43}$$

The product with the fourth column of A_2 implies

$$L_{22}F_{24} + L_{24}(F_{44} - a_f - u) = 0$$

Choosing again $L_{24} = 1$ yields

$$L_{22} = \frac{F_{44} - u - a_f}{F_{24}} = \frac{a_f}{F_{24}} - u = \frac{a_f}{\gamma - 1} - u \tag{A.44}$$

The product with the third column of A_2 implies

$$L_{22}F_{23} - L_{23}a_f + uF_{23}L_{24} = 0$$

or

$$L_{23} = \frac{F_{23}}{F_{24}} = -v = L_{13} \tag{A.45}$$

The column of A_2 pertaining to species s implies

$$L_{22}F_{2k} + uL_{24}F_{2k} - a_f L_{2k} = 0 \ , \qquad k = 4 + s$$

or

$$L_{2k} = \frac{F_{2k}}{F_{24}} = L_{1k} \tag{A.46}$$

The result for multiplication with second column of A_2 is

$$L_{21} + L_{22}(F_{22} - a_f - u) + L_{23}v + L_{24}F_{42} + \sum_{k=5}^{N_e} Y_s L_{2k} = 0 \ , \qquad s = k - 4$$

This can be simplified to

$$L_{21} = V^2 + \frac{a_f}{\gamma - 1}(a_f - u) - \frac{p + \epsilon}{\rho} - \sum_{k=5}^{N_e} Y_s L_{1k} \ , \qquad s = k - 4 \tag{A.47}$$

This can be further simplified to

$$L_{21} = \frac{u}{2}\left(u - \frac{2}{\gamma - 1}a_f\right) + \frac{v^2}{2} \tag{A.48}$$

The left eigenvector L_j for $\lambda_j = u$ where $j = 3, \ldots, N_e$ deserves special attention due to the multiplicity of the root. It will be shown here that it is possible to choose $S + 1$ linearly independent eigenvectors. The full product matrix equation for the eigenvectors L_j is

$$[L_{j1}\ L_{j2}\ \ldots L_{j6}] \begin{bmatrix} -u & 1 & 0 & 0 & 0 & 0 \\ F_{21} & F_{22} - u & (1-\gamma)v & \gamma - 1 & F_{25} & F_{26} \\ -uv & v & 0 & 0 & 0 & 0 \\ F_{41} & F_{42} & uv(1-\gamma) & u(\gamma - 1) & uF_{25} & uF_{26} \\ -uY_1 & Y_1 & 0 & 0 & 0 & 0 \\ -uY_2 & Y_2 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.49}$$

For simplicity only a 6×6 system is shown here. The product corresponding to the fourth column gives

$$L_{j2}F_{24} + L_{j4}(F_{44} - u) = 0$$

or

$$L_{j2} = -uL_{j4} \tag{A.50}$$

It can be shown that the product of the matrices corresponding to the y-momentum equation (column 3) and for any of the species equations (column $k = 4 + s$) yields the same result as the above equation. This is obviously a manifestation of the multiplicity of the eigenvalues. The second column of the product implies

$$L_{j1} + L_{j2}(F_{22} - u) + L_{j3}v + L_{j4}F_{42} + \sum_{k=5}^{N_e} Y_s L_{jk} = 0 \ , \qquad s = k - 4 \tag{A.51}$$

or

$$L_{j1} + L_{j3}v + L_{j4}(F_{42} - uF_{22} + u^2) + \sum_{k=5}^{N_s} Y_s L_{jk} = 0 \ , \qquad s = k - 4 \qquad \text{(A.52)}$$

Similarly the first column gives

$$-uL_{j1} - uvL_{j3} + L_{j4}(F_{41} - uF_{21}) - u\sum_{k=5}^{N_s} Y_s L_{jk} = 0 \ , \qquad k = 4 + s \qquad \text{(A.53)}$$

Multiplying Equation (A.52) by u and adding in the previous equation yields

$$L_{j4}\left(uF_{42} - u^2F_{22} + u^3 + F_{41} - uF_{21}\right) = 0 \qquad \text{(A.54)}$$

It can be shown that the coefficient multiplying the L_{j4} term is zero (even when the specific heats are not constants !) and hence L_{j4} can be chosen arbitrarily. Another way of stating this is that Equations (A.52) and (A.53) are redundant. Thus the eigenvectors corresponding to the multiple roots have to satisfy only two restraints, *viz.* Equations (A.50) and (A.52), and hence the values of L_{jk} for $j \geq 3$ can be chosen arbitrarily.

Associating the continuity equation with $j = 4$ and choosing $L_{43} = 0$, $L_{44} = 1$ and $L_{4k} = 0$ for $k \geq 5$, the other remaining items are given by

$$L_{42} = -u \ , \qquad L_{41} = uF_{22} - F_{42} - u^2 = u^2 - \frac{p + \epsilon}{\rho} \qquad \text{(A.55)}$$

Associating the y-momentum equation with $j = 3$ and choosing $L_{33} = 1$, $L_{34} = 0$ and $L_{3k} = 0$ for $k \geq 5$, the other remaining items are given by

$$L_{31} = -v \ , \qquad L_{32} = 0 \qquad \text{(A.56)}$$

Associating the s^{th} species equation with $j = 4 + s = k$ and choosing $L_{k3} = 0$, $L_{k4} = 0$ and $L_{jk} = \delta_{jk}$ yields the other elements as

$$L_{k1} = -Y_s = -Y_{k-4} \ , \qquad L_{k2} = 0 \qquad \text{(A.57)}$$

In summary the left eigenvector matrix $\mathbf{L}$ for the eigenvalues of F_U is given by

$$\mathbf{L} = \begin{bmatrix} \frac{u}{2}\left(u + \frac{2}{\gamma-1}a_f\right) + \frac{v^2}{2} & -u - \frac{a_f}{\gamma-1} & -v & 1 & \frac{F_{25}}{\gamma-1} & \frac{F_{26}}{\gamma-1} & \dots \\ \frac{u}{2}\left(u - \frac{2}{\gamma-1}a_f\right) + \frac{v^2}{2} & -u + \frac{a_f}{\gamma-1} & -v & 1 & \frac{F_{25}}{\gamma-1} & \frac{F_{26}}{\gamma-1} & \dots \\ -v & 0 & 1 & 0 & 0 & 0 & \dots \\ u^2 - \frac{p+e}{\rho} & -u & 0 & 1 & 0 & 0 & \dots \\ -Y_1 & 0 & 0 & 0 & 1 & 0 & \dots \\ -Y_2 & 0 & 0 & 0 & 0 & 1 & \dots \\ \dots & & \dots & & & & \dots \end{bmatrix} \tag{A.58}$$

It can be easily verified that the *inner* product of any two eigenvalues is non-zero and hence the eigenvectors are non-orthogonal; however, the eigenvectors are linearly independent. One can use the Gram-Schmidt orthonormalization procedure, to make the elements of the set mutually orthogonal.

Appendix B

Considerations for the Computer Code

Since a major fraction of the efforts associated with the current research is algorithm and code development, it is appropriate to point out the important considerations which one must take into account prior to undertaking such a task.

In order to keep the integration procedure independent of the geometry of the individual problems and the specific initial distribution of state vectors, the STAR code requires the allocation of grid points and initial conditions through separate programs. Thus the grid generator and initial-condition generator are kept separate from the spatio-temporal code and these must generate output (file INPUTG.DAT for grid and INPUTD.DAT for initial conditions) in formats consistent with what STAR code demands. In a similar manner the chemistry models are not implemented as separate modules in the code and all the pertinent chemistry data must be supplied in a separate file (INPUTC.DAT for chemistry deck). This increases the robustness of the code in the sense that an arbitrary number of grid topologies, initial conditions and chemical reaction systems may be handled by this code. However, this has the disadvantage that different grid and initial condition generators will be needed for each new kind of geometry and flow conditions, and the maintenance of these small but numerous programs may be confusing.

A numerical code typically produces a large amount of output data. It would be inefficient to store the output after each time-stride, since the overall size of this data will approach gargantuan proportions. For this reason simulations are usually carried out in segments composed of a few hundred time-steps after which the output is dumped out. For the current approach the output can be produced when a selected number of

time-strides have been completed or when the time exceeds a user specified value. The user also specifies the maximum *size* of the time-strides to be used through parametric input. For problems in which interest is limited to the steady state, local time-stepping can be selected in which case the size of the time-strides is irrelevant.

The usage of data pertaining to a selected number of time-stations implies that the temporal states after each time-stride need not be saved even for unsteady flow problems. This means that the storage of state vectors should only provide spatial variations and that only current values in time need be remembered at each spatial node. This has the advantage of curtailing the demands on the CPU memory while in the execution mode. The disadvantage is that a *continuous* motion picture, which may provide insights in the dynamics of fluid motion, cannot be produced . However, the output from various simulations, pertaining to different time-stations, can be organized in a sequential manner and a *discontinuous* motion picture is realizable and can provide valuable information. The storage of output pertaining to various time-stations could itself be very large and in fact exceed the limits of disk storage. For this reason, the long-term storage of these simulations should always be restricted to personal devices such as magnetic tapes or mountable disks.

A software facility should also have a restart capability which utilizes the output dump of a previous calculation. One important consideration for restart cases is that certain parameters be allowed to change in the newer simulation. For example, one may decide to freeze the collapsing of grids in one run and only allow grid sub-division, whereas in a later case both of these procedures might be applied.

Another consideration in building a robust computer software is the *modularization* of individual physical processes which are segmented as individual sub-routines or procedures. Models must be built so that each of these processes is calculated accurately and calibrated separately before the final assembly. For example, in the STAR code, the procedures for grid division and merger constitute two separate sub-routines. Similarly the integration calculations, boundary condition evaluations and the process of updating are done in separate routines. In addition to providing a reasonable organization

for solving the overall problem, the modular approach allows the use of best numerical techniques for each aspect of the problems. This approach is extended further in the STAR code in the form of making available a number of alternative subroutines. For example, two kinds of integration routines are provided and the choice depends upon whether one wishes to perform inviscid or viscous calculations. The calling names and arguments of the two routines are identical, though stored in files with different names, and these can be discriminated at link time. Although a single subroutine could have been written to accomplish both the objectives through logical statements, this was not done in favor of keeping the routines simpler and efficient. Other procedures which perform slightly different calculations are stored and organized in a similar manner. The direct consequence of this approach is that the overall size of the software becomes very large, but only a selected number of routines are linked together to yield a particular simulation. The availability of a large number of alternative routines also has some disadvantages. When a change is made in one of the subroutines to account for something in an efficient manner, this change is typically needed in other similar routines. The situation also demands that the operator of the software be familiar with the function, advantages and disadvantages, and applicability of individual routines in differing situations. However, this is not a serious disadvantage, since it is usually a mistake to consider the software operator an irrelevant intermediary who feeds the computer. The operator should really be an expert who understands the overall organization of the software and should be bold and competent enough to make necessary changes if the need arises. Another aspect of *structured programming* [108] is that helpful comments be provided for each procedure for those who use the software. About 50% of the STAR code consists of comment lines.

One very crucial consideration in the programming of *unstructured* grid codes (not to be confused with structured programming) is the ability to detect the incursion of errors in the pointer system or data-structure manipulations. These errors typically occur after the application of procedures which divide or fuse the cells. It would be inefficient to globally check the assignment of pointers after each change in data-structure of the grids. However, this might be often needed during the earlier stages of the development of the software. With this aspect in sight special *debug* routines have been written for

the STAR code which examine the data-structure on a global basis for possible errors. These routines pin-point the positions (cells, nodes, boundary points, *etc.*,) where there are inconsistencies in the pointer system and provide an output dump of the pointer system while highlighting the regions of inconsistencies. Once the software is thoroughly tested and debugged these routines can be removed from the calling sequences. Since, to err is human, a possibility of errors under special pathological cases always exists and such routines should never be completely discarded, irrespective of the confidence in the software. Such routines should be added into the software at appropriate places if something unexpected happens during a program execution. In the final version of the STAR code the debug routines only scan the initial data for each start or restart case and do not allow further execution of the program if inconsistencies are discovered.

The portability of the computer software should always be taken into account. The STAR code is completely written in FORTRAN, since this language is appropriate for number crunching and it is widely accepted by the scientific community. In order to increase the portability of the software only generic names are used for functions and special capabilities of certain computers in optimizing codes is sacrificed in favor of portability considerations. The STAR code runs on three machines with minor changes (VAX/VMS, ALLIANT and CYBER 205). Fortunately the vectorization directives appear only as comments for the scalar machines and hence the same code can be used on each. The sections of code which are absolutely essential and different for the several computers are added in *utility* routines and are seldom changed. For ease of recognizing the pertinent statements, all lines for other computers also appear as comments in the utility routines. Hence it would be necessary only to "uncommented" a few lines to apply the routines to other systems. The utility routines make system calls (*e.g.*, CPU time evaluations for a given procedure) and do special mathematical operations (*e.g.*, inverting a matrix). Other routines which must be changed on different computers are those which include the `INCLUDE` instruction. Most software has been debugged and tested on the micro-VAX-II, and changes are made and tested only on that machine. Whenever a new change has been made for a code segment (among about a 100 files) which constituted a given simulation, all the pertinent subroutines have been added to a single file through an editor via a command procedure. This procedure could be run

either in interactive or batch (background) mode, without the user having to manually do the overall assembly. The INCLUDE statements in the single big file are then changed through another command procedure for use on a different computer. Once all changes are made, the single file is transferred to the other computer. The advantage of this relatively complicated approach is that it allows nearly the same version of the code on different operating systems for a given simulation.

Appendix C

Data Structure

This appendix is devoted to data structure and necessarily involves a considerable amount of computer mnemonic. The importance of data structure stems from a grid adaptation concept and understanding its logic is essential in implementing an improved and efficient algorithm. The pointer system itself can be subdivided to handle spatial, temporal and chemistry parts of the coding. The procedures of grid division, fusion and extension are detailed in terms of this pointer system.

C.1 Spatial Data Structure

The spatial data structure utilized in this section follows the pointer system as proposed by Dannenhoffer [33].

C.1.1 Cell-to-node Array

Connectivity arrays define the objects to be gridded. The cell-to-node array is defined by `ICELG2(1:10,1:MCELG2)` and indicates the linkage of a given cell to its nodes and parent cell. The colon notation indicates the bounds of validity of this array. Here `MCELG2` denotes the maximum allowable number of cells. In `MCELG2`, the first letter `M` stands for maximum, the letters `CEL` for cells and `G2` to indicate 2-D grid. The current total number of cells is denoted by `NCELG2`. The notation of other arrays is defined along similar patterns. The first nine entries of this array, for a given cell `IC` between 1 and `NCELG2`, point to the nodes of this cell as indicated by the numbering scheme shown

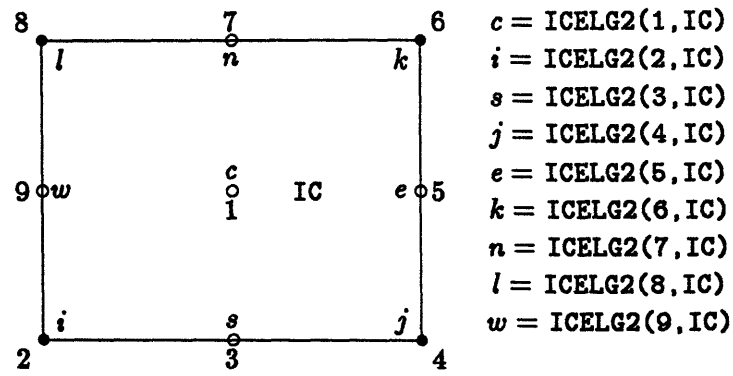

Figure C.1: Node pointers for a given cell IC.

in Figure (C.1). The filled circles denote corner nodes which are always present while empty circles correspond to nodes which may or may not exist. Any node which does not exist is entered as zero pointer. For example the assignment of nodes for cells C and E in Figure (3.5) is

$$
\begin{aligned}
&\texttt{ICELG2(2,}C\texttt{)} = i > 0 \\
&\texttt{ICELG2(3,}C\texttt{)} = s = 0 \\
&\texttt{ICELG2(4,}C\texttt{)} = e > 0 \\
&\ldots \qquad\qquad\qquad\quad \ldots \\
&\texttt{ICELG2(q,}E\texttt{)} = 0 \;, \quad \texttt{q} = 1,3,5,7,9 \\
&\texttt{ICELG2(8,}C\texttt{)} = e > 0 \\
&\ldots \qquad\qquad\qquad\quad \ldots
\end{aligned}
$$

Note that a cell with a non-zero center node is always a divided cell, and therefore is a parent or supercell of four unique children cells. The center node pointer is irrelevant for unsteady flow calculations since a divided cell is not involved in the integration calculations; however, for steady-state situations, in which a multiple-grid may be used, this node-pointer becomes important. In the STAR code the center node pointer is retained to maintain generality and to allow a simple discrimination basis between divided and undivided cells. Note that the cell number IC of a particular cell after sub-division still represents the old cell and its corner pointers do not have to be readjusted. For un-

steady flows, without a multiple grid technique, this means a retention of unnecessary cell numbers, and hence a *linked list* of only undivided cells must be maintained for an efficient integration procedure. A division of a single cell in the domain always increases the total number of cells by four and the opposite holds for the fusion of cells. Hence after the completion of a spatial adaptation cycle the total number of cells differ by factors of four compared to those at the beginning of the cycle.

The tenth element of the cell-to-node array pointer for a given cell IC, namely, ICELG2(10,IC) indicates the supercell or parent of that cell. For the base grid cells (spatial level 0) these pointers are assigned zero values whereas for any finer level (cells embedded once are at level 1 and so on) cells these point positive cell values. Thus if the supercell of cells E and F in Figure (3.5) is denoted by G then

$$\texttt{ICELG2(10,}E\texttt{)} = \texttt{ICELG2(10,}F\texttt{)} = G$$

If this pointer exists for a given cell then there should be exactly three more cells with the same supercell pointer. The supercell pointers are used to avoid expansive search procedure when collapsing of cells is desired.

The importance of consistency checks to data-base structure was pointed out in the previous appendix. The following consistencies should exist for the cell-to-node array and some or all of them should be periodically checked to avoid incurrence of errors:

- ICELG2(j,IC),j=1,9 should be integers between 0 and NNODG2, where NNODG2 denotes the total number of nodes in the domain. Furthermore for a given cell all of the non-zero pointers should be unique.
- ICELG2(j,IC),j=2,4,6,8 should always be strictly positive integers.
- If a cell has non-zero pointers for ICELG2(j,IC),j=3,5,7,9, then it should be a divided cell.
- ICELG2(10,IC) should be an integer between 0 and NCELG2; if non-zero, the cell IC should be at a finer level than the base grid.
- Each divided cell should be a supercell of exactly four cells.

- The north-east node of a given cell should be a south-west node of a neighboring cell — except near boundaries. Similar permutations apply for other nodes.

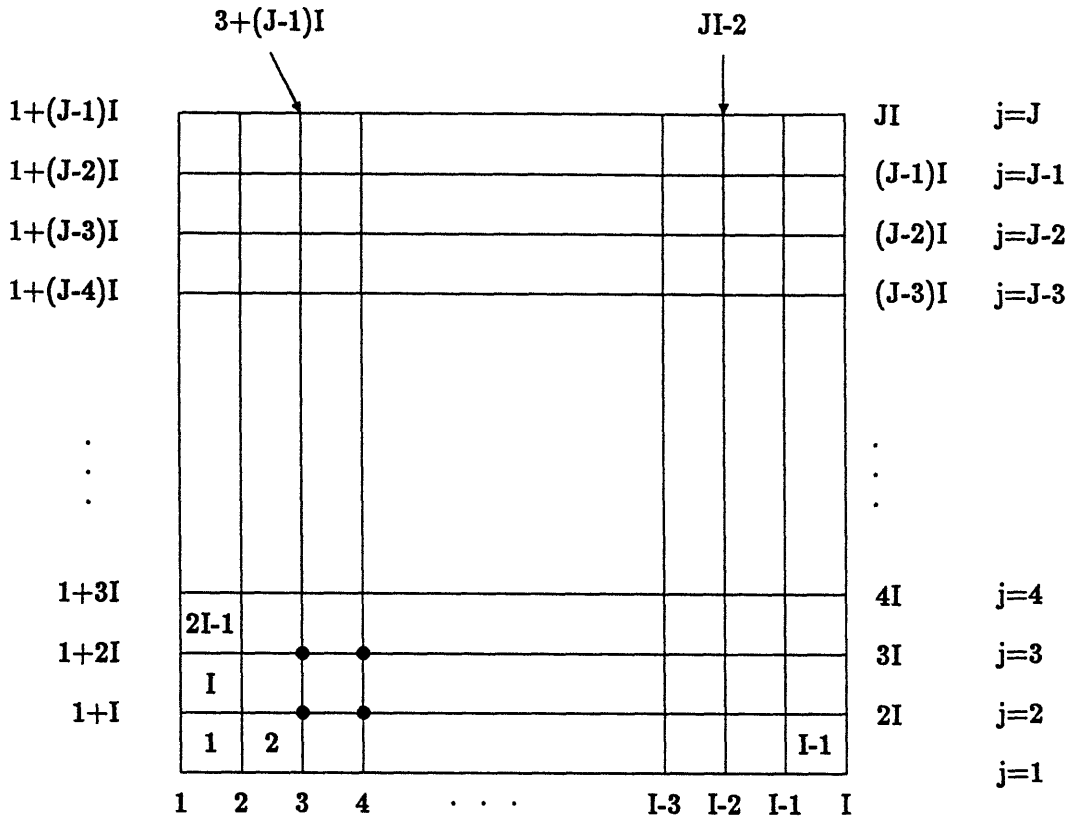

Figure C.2: Initialization of grid pointers for an initial structured grid.

The initial coarsest grid in the STAR code can be generated by considering either the computational domain to be a logical rectangle (structured initial grid) for simple geometries or by an interactive block grid generator for solid bodies embedded in the computational domain.

The initialization of the connectivity arrays can be better explained by the simple grid generator, instead of block grid generator. The interested reader can examine the listing `GNBLOC` of the block grid generator to see how this initialization is done. For

the logical grid the code first reads the geometry at the boundary nodes and creates an algebraic grid for the interior nodes. Consider Figure (C.2) in which the number of nodes along x-axis is I and that along the y-axis is J. The node numbers along the southern edge are numbered $1, 2, \cdots, I$ and the numbering continues in the same fashion for all the rows of the grid. Hence the western face nodes of the logical rectangle are given by $1+(j-1)I$ whereas the eastern face nodes are given by jI for $j \in [1, J]$. The cell numbers are traversed in the same manner, thus the cells adjacent to the western face are numbered as $1+(j-1)(I-1)$ for $j \in [1, J-1]$ and the total number of cells is $(I-1)(J-1)$. Consider the initialization of the cell marked by circles (*i.e.*, cell number $I+1$) with the pointers

```
ICELG2(2,I+1) = I+2
ICELG2(4,I+1) = I+3
ICELG2(6,I+1) = 2I+3
ICELG2(8,I+1) = 2I+2
ICELG2(q,I+1) = 0 , q=1,3,5,7,9,10
```

On an overall basis, the non-zero pointers of all the cells can be assigned by the following sample code:

```
NCELG2 = 0
DO JP = 1, J
   DO IP = 1, I
      NCELG2           = NCELG2 + 1
      ICELG2(2,NCELG2) = IP      + (J-1)*I
      ICELG2(4,NCELG2) = IP + 1  + (J-1)*I
      ICELG2(6,NCELG2) = IP + 1  +     J*I
      ICELG2(8,NCELG2) = IP      +     J*I
   ENDDO
ENDDO
```

Although the STAR code conforms to ANSI standards for FORTRAN, the extension

of VAX-11 FORTRAN are used here to avoid *tame-labels* [108] associated with the CONTINUE statements for the examples in this thesis. The FORTRAN rule of assignment to a real or integer variables can be assumed to be valid for most examples and the exceptions are pointed out if such a need arises. Note that once the adaptive procedure is invoked, the initial structured grid loses its structure.

C.1.2 Node-to-cell Array

This type of array specifies the cells surrounding a given node. The form of this array is NEIBG2(1:4,1:MNODG2), where MNODG2 is the maximum allowable number of nodes. For a given node IN in between 1 and NCELG2, the values NEIBG2(q,IN) point to the south-west, south-east, north-east and north-west cells respectively for q=1,2,3 and 4. Hence for node i in Figure (3.5) the assignments are

$$\texttt{NEIBG2(q,}i\texttt{)} = A, B, C, D \qquad \text{for} \qquad \texttt{q} = 1, 2, 3, 4$$

This array can be used to identify various kinds of spatial interfaces where a grid abruptly changes; for example, consider the interface *j-e-k* of Figure (3.5) — the node-to-cell array of the middle edge node is

$$\texttt{NEIBG2(q,}e\texttt{)} = C, E, F, C \qquad \text{for} \qquad \texttt{q} = 1, 2, 3, 4$$

that is,

$$\texttt{NEIBG2(1,}e\texttt{)} = \texttt{NEIBG2(4,}e\texttt{)}$$

For nodes on a physical boundary some pointers will be zero; for example, for nodes IN on a southern boundary

$$\texttt{NEIBG2(1,IN)} = \texttt{NEIBG2(2,IN)} = 0$$

Thus if all four pointers of a given node are non-zero and unique, it is a common interior node; however, if the four pointers are non-zero and non-unique then the node is an interior middle edge node of a spatial interface. For the present code the distinction between various types of spatial interfaces is not needed, but this information is available

as a by-product if the node-to-cell array assignments are carried out in a manner as prescribed above.

A number of consistency checks can be made for this array, the most important of which is that NEIBG2 be the inverse of ICELG2. For example for all the *interior* nodes IN the following relation should exist

$$\texttt{IN} = \texttt{ICELG2(2,NEIBG2(3,IN))}$$

or inversely for *any* given cell IC

$$\texttt{IC} = \texttt{NEIBG2(3,ICELG2(2,IC))}$$

Both the arrays ICELG2 and NEIBG2 make the process of cell division and fusion extremely efficient since no search involving the neighboring objects is then needed.

The initialization of the node-to cell array can be accomplished for the structured grid of Figure (C.2) after the initialization of the cell-to-node array by the following sample code:

```
DO IC = 1, NCELG2
   NEIBG2(1,ICELG2(6,IC)) = IC
   NEIBG2(2,ICELG2(8,IC)) = IC
   NEIBG2(3,ICELG2(2,IC)) = IC
   NEIBG2(4,ICELG2(4,IC)) = IC
ENDDO
```

C.1.3 Node-Arrays

The node-arrays contain the geometry information, state vectors and some other variables at all of the computational nodes. The geometry of the physical domain is specified by the array GEOMG2(1:2,1:MNODG2). For a given node IN between 1 and NNODG2 the x and y-coordinates of the node are given by

$$\texttt{GEOMG2(q,IN)} = x \text{ or } y\text{-coordinate of IN for q = 1 or 2}$$

The state vector array is defined by DPENG2(1:MEQNFL,1:MNODG2), where MEQNFL is the maximum allowable number of equations to be solved. The current number of dependent variables is denoted by NEQNFL and is a constant for a given case. As an example, the fifth dependent variable, which is the product of local global density and mass fraction of the first species, is specified by DPENG2(5,IN).

Other node arrays include the pressure PRESG2(IN), temperature TEMPG2(IN), artificial viscosity coefficient SIGGG2(IN) and residual change values CHNGE2(1:MEQNFL,IN) at a node IN between 1 and NNODG2.

C.1.4 Cell-Arrays

The cell-arrays hold information pertaining to some or all of the cells in the computational domain. The list of all undivided cells is defined by ICELA2(1:MCELG2). In consistency with the previous notation NCELA2 denotes the current total number of undivided cells. This list is useful for integration purposes and for division and collapse routines, since only the undivided cells are integrated for unsteady flows and only these cells can be further divided or fused to yield an earlier supercell. As mentioned earlier, the detection of undivided cells is accomplished by examining the center node of each cell in the domain. The following sample code can be used to accomplish this:

```
NCELA2 = 0
DO IC = 1, NCELG2
   IF (ICELG2(1,IC) .NE. 0) THEN
      NCELA2          = NCELA2 + 1
      ICELA2(NCELA2) = IC
   ENDIF
ENDDO
```

Note that for those cases for which spatial adaptation procedure is frozen at all times, this list degenerates to

$$\texttt{ICELA2(1:NCELG2)} = \texttt{1:NCELG2} \qquad (\texttt{NCELA2} = \texttt{NCELG2})$$

This array must be updated after each spatial adaptation cycle which involves a division and fusion of cells, removal of islands and voids, and extension of the region containing the spatially adapted cells.

The arrays MRKCA2(1:MCELG2) and MRKDA2(1:MCELG2) hold the cell numbers which are marked for possible fusion and division respectively. The current total number of such cells is denoted by NCELC and NCELD respectively. Since only a fraction of cells need be fused or divided for a given spatial adaptation cycle, a more frugal maximum dimension of these arrays could have been selected; but since these arrays are also used beyond spatial adaptation procedure the maximum dimension MCELG2 is retained.

The array CHNGA2(1:3,1:MCELG2) holds the information pertaining to spatial differences of three or less criteria variables which are used for local embedding or fusion. These differences are computed for all undivided cells between 1 and NCELA2. The procedure of computations of these differences is discussed in Section (5.3). Although this procedure is applicable for more than three criteria variables, it becomes expensive and inefficient to carry more than two variables.

The cell-arrays MRKCA2, MRKDA2 and CHNGE2 do not need to be initialized for the logical structured grid of Figure (C.2), since these arrays are evaluated anew for each spatial adaptation cycle.

C.1.5 Boundary-Array

This array contains information pertaining to the nodes on the domain boundaries. This is needed to apply boundary conditions, perform interpolation functions and facilitate grid adaptation near the boundaries. The connectivity array for boundary nodes is denoted by IBNDG2(1:5,1:MBNDG2), where MBNDG2 indicates the maximum allowable

number of boundary points and NBNDG2 is the current total number of these points. For a given boundary point IB, the first entry of the array indicates the actual node on the boundary. Similarly the second and third entries indicate the two finest level cells adjacent to the boundary node. The fourth entry indicates the orientation of the boundary; it is 3, 5, 7, 9 if the boundary is south, east, north, west surface respectively. This denotation is consistent with the pointers of the cell-to-node array ICELG2; similar numbers are assigned for the four corner nodes of the logical domain. The fifth entry denotes the type of boundary condition to be applied at the node in question. The types of boundary conditions were discussed in Chapter 7 and more details can be found in the subroutine E2BCNO.

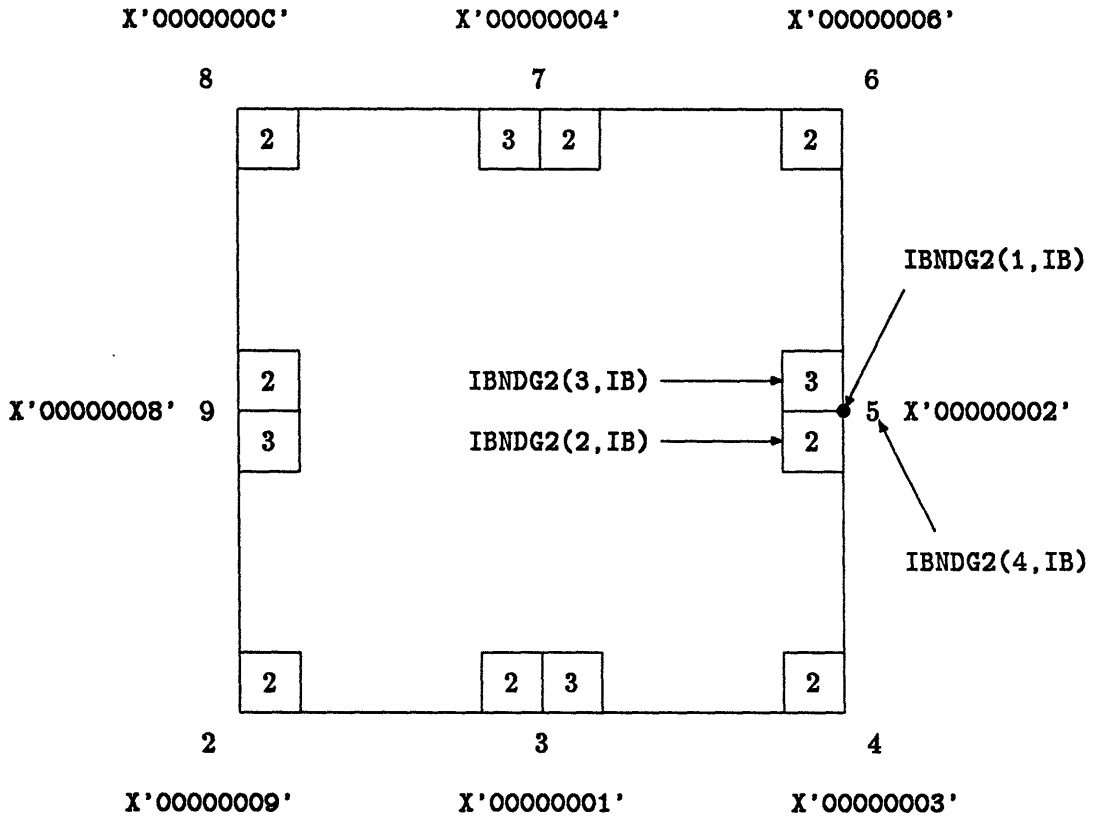

Figure C.3: Initialization of boundary pointers for an initial structured grid.

For the initial grid generator of the logical rectangle in Figure (C.2) the bound-

ary points start at the south-west corner of the domain and traverse the computational domain in the counter clockwise direction. For the southern surface boundary (IB=2,...,I-1)

```
IBNDG2(1,IB) = IB
IBNDG2(2,IB) = IB-1
IBNDG2(3,IB) = IB
IBNDG2(4,IB) = 3
```

Similarly the eastern face of the logical rectangle is given by (IB=I+j-1 for j=2,3,...,J-1)

```
IBNDG2(1,IB) = j*I
IBNDG2(2,IB) = (j-1)*(I-1)
IBNDG2(3,IB) = j*(I-1)
IBNDG2(4,IB) = 5
```

Similar assignments of these pointers is indicated in Figure (C.3). The numbers within the boxes indicate the cells adjacent to the boundary node whereas the numerals outside the computational domain indicate the orientation of the boundary. Note that there is only one cell adjacent to the corner boundary points, *i.e.*, IBNDG2(3,IB)=0 for these locations.

A number of consistency checks can be made for these pointers; however, the important checks must evaluate the consistency of IBNDG2 with ICELG2 and NEIBG2 at the boundary locations.

C.1.6 Auxiliary Pointers

The boundary array discussed earlier provides a connectivity of boundary points to the boundary nodes and cells. An inverse relation is needed for cells near the boundaries for spatial adaptation procedure. Alternatively if a cell can be recognized as a boundary

cell at the time of sub-division or fusion, the boundary-array can be scanned to locate the boundary points which correspond to the cell in question. The second approach is used here since the size of the boundary-array is generally much smaller compared to the total number of nodes or cells and only a small fraction of cells are adjacent to a boundary and hence scanning the array is not expansive. The auxiliary array KAUXG2(1:MCELG2) is used for this purpose. To save on further storage this array is used to hold other information besides the boundary cell details. It has the following hexadecimal form for each byte:

$$\texttt{KAUXG2(IC)} = h_8 h_7 h_6 h_5 h_4 h_3 h_2 h_1$$

The first byte or h_1 is used to indicate that the cell IC is a boundary cell and points out its orientation. If the cell is not adjacent to a boundary this byte is set equal to zero. Figure (C.3) shows the assignment of this byte as hexadecimal values for each orientation. As an example, consider a cell IC on southern boundary. During initialization or when the cell is created from a supercell on the same boundary due to local embedding, this byte can be set by the following statement:

```
KAUXG2(IC) = OR (KAUXG2(IC),X'00000001')
```

Note that this statement only modifies the last byte of the auxiliary array. Also note that the hexadecimal form for a corner cell is obtained by logical addition of the bytes pertaining to the two corresponding boundary surfaces. For example, the north-west corner has the hexadecimal form

```
X'0000000C' = OR (X'00000004',X'00000008')
```

Thus, during the process of cell division or fusion a cell IC can be identified to be a boundary cell if the number

```
KB = AND (KAUXG2(IC),X'0000000F')
```

is non-zero and the type of boundary can be deciphered from the non-zero value of KB.

The second byte or h_2 indicates that the cell was recently divided and hence must not be collapsed. Thus if a cell is marked for possible fusion and h_2 is found to be

non-zero, the process of fusion is delayed. This means that the cells divided in the past few spatial adaptive cycles will not be fused until a specific number of adaptive cycles has elapsed. The details of this pointer will be explained later.

The third byte h_3 indicates the type of boundary interpolation functions to be used for the geometry of the middle edge nodes when a cell on a boundary is locally divided. Depending on this value linear, circular and cubic spline surfaces are considered for interpolation.

The fourth byte indicates the special cells which are never allowed to collapse to form larger cells. This is useful for locations where special features are known to be stationed at all times.

The fifth byte is used to indicate the spatial level of the cells. As pointed out earlier the initial coarse cells are at level 0, the children cells of these cells are at level 1, and so on. Note that the maximum possible level of any cell for this approach can be atmost 15; however this never occurs since the maximum allowable spatial level of the cells, MALVG2, is assigned a value of 6 through a PARAMETER statement. The current maximum level of cells for a particular run is often less than MALVG2 and is denoted by NALVG2. If a given cell LC is sub-divided into four cells with cell numbers IC, IC+1, IC+2, IC+3 then the level pointer can be determined and a possible check for avoiding division can be evaluated by the following sample code:

```
K5LC  = AND ( KAUXG2(LC), X'000F0000' ) ! 5th byte of LC
LEVLC = ISHFT ( K5LC, -16)              ! Level of LC
LEVIC = LEVLC + 1                       ! Level of IC
IF (LEVIC .GT. NALVG2) RETURN           ! Abort division process
K5IC  = K5LC + 2**16                    ! 5th byte of IC
DO J = 0, 3
   KAUXG2(IC+J) = OR ( KAUXG2(IC+J), K5IC )
ENDDO
```

The level pointer calculation will be simply represented by the function call LEVEL(IC)

for cell IC in the subsequent. The level pointer is also used in enforcing other rules for both grid division and fusion and will be discussed further as the need arises. The remaining three bytes are currently not utilized in the STAR code. The description of spatial adaptation procedure can now be explained in terms of the spatial data-structure.

C.2 Temporal Data Structure

The cell-array CELLTI(1:MCELG2) defines the time-step for each undivided cell in the domain. Thus for a cell index IC, between 1 and NCELA2, the cell-time-step for cell number ICELA2(IC) is CELLTI(ICELA2(IC)). A separate array for cell-time-steps is needed to avoid repeating this calculation during various steps within a time-stride. Note that for steady state applications this array may not be required, since the time-step for each cell can be computed at the same time when it is being integrated and the local value (rather than global minimum) may be used.

Since the cells with the same time-step are integrated together a link-list defining the cells with iso-temporal level is needed. The maximum allowable temporal level, defined by a PARAMETER statement, is denoted by MMAXTI, whereas the user-supplied maximum level for a specific run is denoted by NGIVTI. The actual maximum level, NMAXTI, may be less than or equal to the given value and may change its value from one time-stride to next. As an example consider that a programmer has set the value MMAXTI=6 for all the program declarations and a user wants to only use four levels of temporal embedding to avoid temporal level stiffness for a nearly frozen flow calculation, so he sets NGIVTI=3 in the input data file. If during the start of the computations he does not want to use pre-embedding and the cells have nearly the same volumes then the difference between Δt_{max} and Δt_{min} will be very small and he would get the result NMAXTI=0. However, at a later time when he does just one level of spatial adaptation he would get NMAXTI=1.

The link-list for cells with iso-temporal level is defined by the one-dimensional array ICELTI(1:MCELG2). The cells with same temporal levels form contiguous indices of this

array. Thus the first cell at level 0 is given by ICELTI(1), the second cell at level 0 is given by ICELTI(2), and so on. The last cell index at level 0 is indicated by an array value ILVLTI(2,0) and so the last cell at level 0 is given by ICELTI(ILVLTI(2,0)). The index for first cell at level 1 ILVLTI(1,1) is one more than of the last cell at level 0. The general form of the level index pointer is ILVLTI(1:2,0:MMAXTI). Thus the indices for first and last cells at temporal level LNT are respectively given by ILVLTI(1,LNT) and ILVLTI(2,LNT). The assignment of the temporal levels and the creation of the link-list array is detailed in subroutine E2TIMU. The cells within a certain group of levels can be stored in any manner.

As an example consider one integration pass of cells at temporal level LNT as in the following sample code:

```
      DO JCELL = ILVLTI(1,LNT), ILVLTI(2,LNT)

C        Find the cell to be integrated
         ICELL = ICELTI(JCELL)

C        Set up node pointers for this cell
         KSW = ICELG2( 2,ICELL)

         . . .   . . .   . . .   . . .

C        Perform flux balance for this cell

         . . .   . . .   . . .   . . .

      ENDDO
```

The integration procedure for the code is listed in routine E2SOLU and the determination of the integration sequences can be found in the routine TWODOU.

C.3 Chemistry Data Structure

The data structure for chemistry holds information pertaining to the number of species (NSPECH/MSPECH) and the number of reactions (NREACH/MREACH). *Reaction-arrays* like PREFCH(1:MREACH), EXPFCH(1:MREACH), ENEFCH(1:MREACH) contain pre-exponential factor, exponent of temperature and the energy term, respectively, for each forward direction of a reaction. The reaction-array NSRKCH(1:MREACH) contains total number of species in any reaction, this is helpful in avoiding the species with zero stoichiometric coefficients in certain manipulations (see the example below). The *species-arrays* hold the following typical information:

ATWTCH(1:MSPECH) : molecular mass of each species

FMHTCH(1:MSPECH) : heat of formation of a species

SPCPCH(1:MSPECH) : first constant a_s in constant pressure specific heat

SPBSCH(1:MSPECH) : second constant b_s in constant pressure specific heat

YSPECH(1:MSPECH) : free stream or reference mass fractions

YMAXCH(1:MSPECH) : maximum possible mass fractions

In addition to these, there are arrays interconnecting various species among reactions; specific examples being:

IALPCH(IS,IR) : stoichiometric coefficients of species IS on the left side of reaction IR

IBETCH(IS,IR) : stoichiometric coefficients of species IS on the right side of reaction IR

IALOCH(IS,IR) : exponent of species IS concentration in the forward rate of reaction IR

IBETCH(IS,IR) : exponent of species IS concentration in the backward rate of reaction IR

ITABCH(ISP,IR) : table of species numbers, between 1 and NSRKCH(IR), involved in reaction IR

The allocation of the table od species numbers deserves attention. Consider the

example of Rogers and Chinitz model, the reactions are numbered as

$$H_2 + O_2 \rightleftharpoons 2OH \qquad \text{IR} = 1$$
$$H_2 + 2OH \rightleftharpoons 2H_2O \qquad \text{IR} = 2$$

There are five species and these are numbered as O_2, OH, H_2, H_2O, N_2 for IS between 1 and 5 respectively. Thus we note that

```
NSRKCH(1) = NSRKCH(2) = 3
```

and for the first reaction `ITABCH(1,IR) = 3` for H_2, `ITABCH(2,IR) = 1` for O_2, `ITABCH(3,IR) = 2` for OH. Similar allocation holds for the second reaction. This avoids the usage of IF-THEN clauses for the inert species. Thus the source term for a species IS at a node INODE can be easily determined by the following sample code:

```
C
C       DETERMINE TEMPERATURE AT THE GIVEN NODE
C
        T = TEMPG2(INODE)
C
C       COMPUTE THE CONTRIBUTION WREACT TO THE SOURCE TERMS FROM ALL
C       THE REACTIONS

        DO IR = 1, NREACH
           AKFR  = EXP ( PREFCH(IR) + EXPFCH(IR)*LOG(T) - ENEFCH(IR)/T )
           AKBR  = EXP ( PREBCH(IR) + EXPBCH(IR)*LOG(T) - ENEBCH(IR)/T )
           PRODF  = 1.0
           PRODB  = 1.0
           DO IS = 1, NSRKCH(IR)
              ISP   = ITABCH(IS ,IR)
              PRODF = PRODF*DPENG2(4+ISP,INODE)/AMWTCH(ISP)**IALOCH(ISP,IR)
```

```
            PRODB = PRODB*DPENG2(4+ISP,INODE)/AMWTCH(ISP)**IBTOCH(ISP,IR)
          ENDDO
          WREACT(IR) = AKFR*PRODF - AKBR*PRODB
        ENDDO
C
C       COMPUTE THE SOURCE TERMS FOR ALL RELEVANT SPECIES
C
        DO JS = 5, NEQNFL
          IS     = JS - 4
          SUMWT  = 0.
          DO IR = 1, NREACH
            SUMWT  = SUMWT + (IALPCH(IS,IR)-IBETCH(IS,IR))*WREACT(IR)
          ENDDO
          SOURCE(JS) = AMWTCH(IS)*SUMWT
        ENDDO
```

C.4 Grid Division

The cells to be divided are stored in the link-lists MRKDA2. Before a particular cell LC can be divided a number of other conflict rules are examined, and if any hold the division procedure for this particular cell is not carried out. The simplest of these rules examine if there is room in the data base for additional pointers which the newly created cells would demand. This involves

- Check for overflow in node-arrays (NNODG2+5.LE.MNODG2)
- Check for overflow in cell-arrays (NCELG2+4.LE.MCELG2)
- Check for overflow in boundary pointers (NBNDG2+2.LE.MBNDG2)

Next the spatial level of the cell LC to be divided is examined by

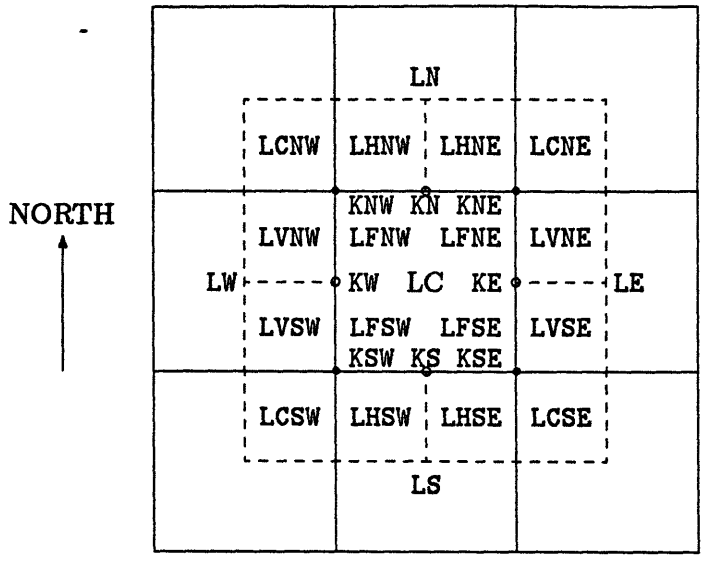

Figure C.4: Nodes and cells bordering cell LC.

```
LEVLC = LEVEL (LC)
```

and if this level is greater than or equal to a user-specified maximum level, the division process is not carried out. The nodes corresponding to the cell LC are determined by the cell-to-node array and once known the node-to-cell array is used to determine the cells neighboring LC. The situation is depicted in Figure (C.4); the node and cell numbers are signified to begin with the letters K and L respectively, and the assignments indicate compass point directions. The nodes marked by open circles may not exist, in which case the corresponding neighboring cells in the dashed boxes do not exist, and the cell numbers then correspond to the coarser level. The levels of cells LCSW, LHSW, LCSE, LVSE, LCNE, LHNE, LCNW, LVNW are evaluated and division process is aborted if the difference of any spatial level and that of LC is less than 0 or greater than 1. For the level pointer rule consider the three possibilities for cells LHSW and LC as indicated in Figure (C.5). The division is not allowed to occur in the last case. The integer LEVDIF indicates the difference between the two cell levels, *i.e.*,

LEVDIF = LEVEL(LHSW) - LEVEL(LC)

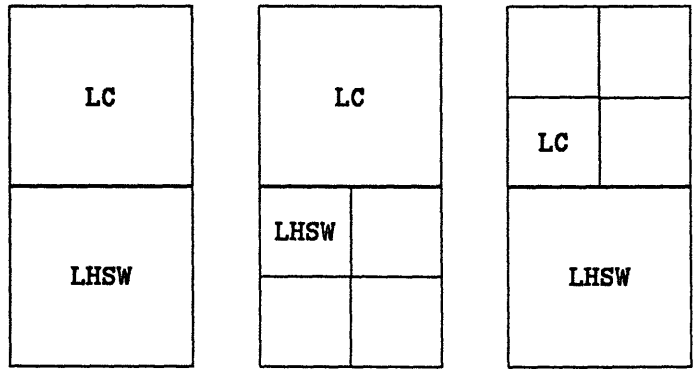

Figure C.5: Three possible situations for spatial level differences.

After all preliminary tests are performed, the cell LC is ready to be divided. First a node is created at its centroid with dependent variables equal to the average values at corner nodes. Next the nodes KS, KE, KN, KW at the face midpoints are created if such nodes do not already exist and new nodal values for the node-arrays are assigned as the average values of the corresponding corner nodes. The node-to-cell arrays are adjusted appropriately to account for additional nodes. As an example consider the following sample code that assigns values at the southern node:

```
C
C       Does southern node already exist; if not create it
C
        IF (KS .EQ. 0) THEN
C
C           Increase total number of nodes by 1
            NNODG2        = NNODG2 + 1
```

```
C          Assign last node to southern node
          -KS           = NNODG2
C          Adjust node-arrays
           GEOMG2(1,KS) = 0.5*(GEOMG2(1,KSW)+GEOMG2(1,KSE))
C
C          See if the southern edge is a boundary
C
           IF (LHSW .NE. 0 .AND. LHSE .NE. 0) THEN
              NEIBG2(1,KS) = LHSW
              NEIBG2(2,KS) = LHSE
           ELSE
              NEIBG2(1,KS) = 0
              NEIBG2(2,KS) = 0
           ENDIF
C
       ENDIF
```

Only one example of a node-array has been shown here; the conditional IF-THEN structure is needed for southern external boundary and internal boundaries due to embedded solid objects. Next the cell-to-node array pointers are created for new nodes, *i.e.*,

```
ICELG2(j,LC) = KC, KS, KE, KN, KW      for  j = 1,3,5,7,9
```

Four new fine cells are created as

```
LFSW   = NCELG2 + 1
LFSE   = LFSW   + 1
LFNE   = LFSE   + 1
LFNW   = LFNE   + 1
NCELG2 = NCELG2 + 4
```

The cell numbers for the subcells of a given cell are always related in the same manner, *i.e.*, the south-west cell has the least cell number, south-east cell number is one more than this, and so on. These relative differences remain the same even when the actual cell numbers change after other grid alterations. Next the cell-to-node pointers are initialized for the new subcells. For example, the non-zero pointers for LFSW are

```
ICELG2(j,LFSW) = KSW, KS, KC, KW, LC      for  j = 2,4,6,8,10
```

The auxiliary pointer for this cell is initialized as

```
KAUXG2(LFSW) = K5LFSW + 48
```

where K5LFSW is the fifth byte of the cell LC with a unit value incremented to it, i.e.,

```
K5LFSW = AND (KAUXG2(LC), X'000F0000) + 16**4
```

to indicate that the spatial level of the new cells is one more than that of the parent cell. The integer 48 increments the second byte of the auxiliary pointer by 3 indicating that the newly created cells can not be collapsed for three more spatial adaptation cycles. Each new spatial adaptation cycle after the current one will reduce the number in the second byte by unit (decimal number 16) until this byte becomes zero. The cell-to-node pointers of the neighboring cells are adjusted to account for newly created nodes; for example for the southern node

```
IF (LHSW.NE.0 .AND. LHSW.EQ.LHSE) ICELG2 (7,LHSW) = KS
```

The node-to-cell pointers of all the remaining newly created nodes are then adjusted. For example for center and south nodes

```
NEIBG2(j,KC)  = LFSW, LFSE, LFNE, LFNW    for  j = 1,2,3,4
NEIBG2(3,KSW) = LFSW
```

```
NEIBG2(j,KS ) = LFSE, LFSW                    for  j = 3,4
NEIBG2(4,KSE) = LFSE
```

If LC is along a boundary then the boundary pointers must be adjusted. The cell is aligned with a boundary surface if the pointer

```
KB = AND (KAUXG2(LC),X'0000000F')
```

is non-zero. For example if KB is 1 then the cell is aligned with a southern boundary. Next the boundary pointers IB1 and IB2, corresponding to the corner boundary nodes of LC, are determined by scanning the whole boundary pointer array for which the following statements hold

```
IBNDG2(3,IB1) = LC
IBNDG2(2,IB2) = LC
```

An additional corner boundary point IB3 is required if the cell LC is aligned with two boundary surfaces simultaneously. If the parent cell is deciphered to be on a single southern boundary, for example, the boundary pointers are adjusted as

```
C
C       DIVIDED CELL WAS ALONG SOUTHERN EDGE  (KB = 1)
C
C
        NBNDG2           = NBNDG2 + 1     ! Increase boundary pointers by 1
        IBNDG2(1,NBNDG2) = KS             ! Node for new boundary pointer
        IBNDG2(2,NBNDG2) = LFSW           ! First  cell for this pointer
        IBNDG2(3,NBNDG2) = LFSE           ! Second cell for this pointer
        IBNDG2(4,NBNDG2) = 3              ! Surface orientation
        IBNDG2(5,NBNDG2) = IBNDG2(5,IB1) ! B.C. type
        IBNDG2(2,IB2)    = LFSE           ! First  cell adjacent to IB2
```

```
      IBNDG2(3,IB1)    = LFSW          ! Second cell adjacent to IB1
C        -
C     Correct first byte of boundary pointers for appropriate fine cells
C
      KAUXG2(LFSW)     = OR (KAUXG2(LFSW),X'00000001')
      KAUXG2(LFSE)     = OR (KAUXG2(LFSE),X'00000001')
```

The boundary pointers also are examined to see if special interpolation functions are needed to define the geometry at the middle edge node that conforms to a special solid boundary surface. For example, a quadratic form may be used for a circular arc bump and a cubic spline for other surfaces. Further details on the division process may be found in the subroutine G2DIVO that appears in Appendix D.

C.5 Grid Collapse

The cells to be fused are stored in the link-lists MRKCA2. For a given cell number which is pointed to by this link-list, other cells in the list are examined to see if three additional cells with the same non-zero supercells have been flagged for fusion. The merger occurs only when all four subcells of a previously divided cell are so tagged to be fused. Once located, the cells are arranged according to increasing cell numbers so that LFSW, LFSE, LFNE, LFNW can be determined. Figure (C.4) contains the notation of cells and nodes. The fine cells in the MRKCA2 list are contiguous in the sense that LFNE = LFSE + 1, *etc.*, and that the supercell LC is given by

```
LC = ICELG2(10,LFSW)
```

In order to avoid spatial level stiffness, the levels of those cells neighboring LC are examined, and if the difference between neighboring cell levels and that for LC is either less than 0 or greater than 1, the fusion process is aborted. Actually only corner cells LCSW, LCSE, LCNE, LCNW need be evaluated in this way, edge neighbor cells can be

examined to see if middle fine cells are divided. For example, the southern edge cells satisfy the spatial level restriction if the following conditions are not met

```
IF (ICELG2(3,LFSW) .NE. 0) RETURN
IF (ICELG2(3,LFSE) .NE. 0) RETURN
```

If the supercell LC is permanently marked to reside in a pre-determined region of spatial resolution, then the collapse process is aborted. This is accomplished by

```
IF ( AND(KAUXG2(LC), X'0000F000') .NE. 0 ) RETURN
```

After all preliminary tests are performed, the cells are ready to be fused. First the center node of the cell LC is flagged for removal. In the collapse procedure only the cell numbers are altered, while the node numbers (even those to be deleted) are retained unaltered. The flagged nodes are removed simultaneously once the fusion process for all cells in the link-list is completed. The side nodes of LC are flagged in a similar manner if these are no longer needed by the neighboring cells. As an example consider the possibility of flagging node KS

```
C
C       Mark southern node for deletion if need be
C
        IF (LHSW .EQ. LHSE) THEN
C
C           First examine "interior" node
C
            DPENG2(1,KS) = -99.    ! Flag this node
            KSS          = KS      ! Save node value
            KS           = 0       ! Delete node locally in routine
            LS           = LHSW    ! Southern cell is a single cell
C
```

```
      ELSE IF (LHSW .EQ. 0 .OR. LHSE .EQ. 0) THEN
C
C         Now examine node on internal or external boundary
C
          DPENG2(1,KS) = -99.    ! Flag this node
          KSS          = KS      ! Save node value
          KS           = 0       ! Delete node locally in routine
          LS           = 0       ! Southern cell non-existent
      ELSE
C
C         Southern cells are different and so node KS is needed
C
          LS  = ICELG2(10,LHSW) ! Southern larger cell
C
      ENDIF
```

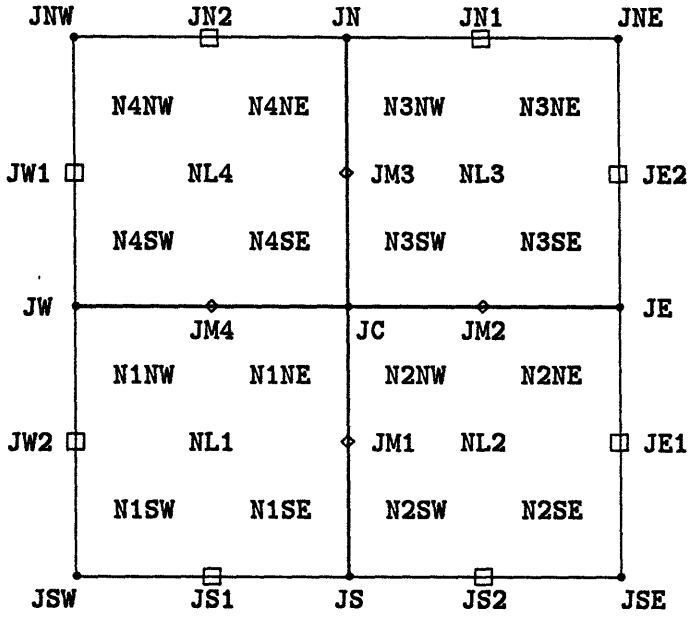

Figure C.6: Pointers associated with the last four cells in the domain.

Since the cell fusion procedure would otherwise create gaps in cell numbers for the fine cells in the cell-arrays, such cells are replaced by the last four cell numbers in the domain. These cells are given by

```
NL4 = NCELG2      !  Interchange LFNW with NL4
NL3 = NL4 - 1     !  Interchange LFNE with NL3
NL2 = NL3 - 1     !  Interchange LFSE with NL2
NL1 = NL2 - 1     !  Interchange LFSW with NL1
```

The pointers of these four cells are indicated in Figure (C.6); the situation is complicated because one or more of the last four cells may be locally divided. The nodes are marked by the labels with first letter J whereas the cells are indicated by the letter N. The nodes marked by solid circles always exist, whereas the nodes marked by diamonds or boxes may or may not exist. Note that in particular if NL1 is undivided then N1SW, N1SE, N1NE, N1NW have the same cell number as NL1; but the corresponding edge nodes of this cell may still exist since there is a possibility of division of a neighboring cell. The cell-to-node pointers of the interchanged cells are adjusted according to the following sample code:

```
DO JP = 0, 3
   DO IP = 1, 10                          ! Update nodes & supercell
      ICELG2(IP,LFSW+JP) = ICELG2(IP,NL1+JP)
   ENDDO
   KAUXG2(LFSW+JP) = KAUXG2(NL1+JP)   ! Update aux. pointers
ENDDO
```

Other cell-arrays are adjusted similarly. If any of the last four cells is divided, some JM nodes marked by diamonds will be non-zero and their pointers will require adjustment. For example consider the case when NL1 may be divided

```
JM1 = 0  ! Initialize middle edge nodes
```

```
      JM4 = 0
      IF (ICELG2(1,NL1) .NE. 0) THEN
         JM4 = ICELG2(7,NL1)
         JM1 = ICELG2(5,NL1)
      ENDIF
      . . . . . .

      IF (JM1 .NE. 0) THEN
         IF (NEIBG2(1,JM1) .EQ. NL1) NEIBG2(1,JM1) = LFSW
         IF (NEIBG2(4,JM1) .EQ. NL1) NEIBG2(4,JM1) = LFSW
         IF (NEIBG2(2,JM1) .EQ. NL2) NEIBG2(2,JM1) = LFSE
         IF (NEIBG2(3,JM1) .EQ. NL2) NEIBG2(3,JM1) = LFSE
      ENDIF
      . . . . . .
```

The node-to-cell pointers of the interchanged cells are adjusted as

```
      IF (NEIBG2(3,JSW) .EQ. NL1) NEIBG2(3,JSW) = LFSW
      IF (NEIBG2(4,JS ) .EQ. NL1) NEIBG2(4,JS ) = LFSW
      IF (NEIBG2(1,JC ) .EQ. NL1) NEIBG2(1,JC ) = LFSW
      IF (NEIBG2(2,JW ) .EQ. NL1) NEIBG2(2,JW ) = LFSW
      . . . . . .
C
C     Update the other non-zero middle edges
C
      IF (JS1 .NE. 0) THEN
         IF (NEIBG2(3,JS1) .EQ. NL1) NEIBG2(3,JS1) = LFSW
         IF (NEIBG2(4,JS1) .EQ. NL1) NEIBG2(4,JS1) = LFSW
      ENDIF
      . . . . . .
```

If any of the last four cells is divided, then it is the supercell of some other cells NSONJ and its supercell will have to be updated. This can be accomplished by

```
      IF (ICELG2(1,NL1) .NE. 0 .OR. ICELG2(1,NL2) .NE. 0 .OR.
     1    ICELG2(1,NL3) .NE. 0 .OR. ICELG2(1,NL4) .NE. 0) THEN
          DO  NSONJ = 1, NCELG2
            ISUP = ICELG2(10,NSONJ)
            IF (ISUP .GE. NL1) THEN
              IF (ISUP .EQ. NL1) ICELG2(10,NSONJ) = LFSW
              IF (ISUP .EQ. NL2) ICELG2(10,NSONJ) = LFSE
              IF (ISUP .EQ. NL3) ICELG2(10,NSONJ) = LFNE
              IF (ISUP .EQ. NL4) ICELG2(10,NSONJ) = LFNW
            ENDIF
          ENDDO
      ENDIF
```

The boundary pointers which point to the interchanged cells have to be adjusted, for example

```
      IF (AND(KAUXG2(NL1),X'0000000F') .NE. 0 .OR.
     1    AND(KAUXG2(NL2),X'0000000F') .NE. 0 .OR.
     2    AND(KAUXG2(NL3),X'0000000F') .NE. 0 .OR.
     3    AND(KAUXG2(NL4),X'0000000F') .NE. 0     ) THEN
          DO  IP = 2, 3
             DO  IB = 1, NBNDG2
               IF (IBNDG2(IP,IB) .GE. NL1)
     1            IBNDG2(IP,IB) = LFSW + IBNDG2(IP,IB) - NL1
             ENDDO
          ENDDO
      ENDIF
```

The node-to-cell pointers of the supercell nodes are adjusted as

```
        NEIBG2(j,KC) = 0        for  j = 1,2,3,4
        NEIBG2(3,KSW) = LCELL
        NEIBG2(4,KSE) = LCELL
        NEIBG2(1,KNE) = LCELL
        NEIBG2(2,KNW) = LCELL
C
        IF (KS .NE. 0) THEN
          NEIBG2(3,KS)  = LCELL
          NEIBG2(4,KS)  = LCELL
        ELSE
          NEIBG2(j,KSS) = 0    for  j = 1,2,3,4
        ENDIF
        . . . . . .
```

The supercell pointers must be adjusted and the node pointers of the larger neighbor cells must be adjusted

```
        NCELG2 = NCELG2 - 4   ! Adjust total number of cells
        ICELG2(j,LCELL) = 0, KS, KE, KN, KW   for j = 1,3,5,7,9
C
C       Reset edge node pointers of all neighboring cells
        IF (LS .NE. 0) ICELG2(7,LS) = KS
        IF (LE .NE. 0) ICELG2(9,LE) = KE
        IF (LN .NE. 0) ICELG2(3,LN) = KN
        IF (LW .NE. 0) ICELG2(5,LW) = KW
```

If LC happens to be on a boundary then the boundary pointers IB1, IB2, IB3 corresponding to the surface nodes are determined and the pointers are adjusted appropriately. For example if LC is aligned with the southern boundary then these boundary pointers are determined and adjusted as

```
        IB1 = 0       ! Initialization
```

```
IB2 = 0
IB3 = 0
DO  IB = 1, NBNDG2
  IF (IBNDG2(1,IB) .EQ. KSW) IB1 = IB
  IF (IBNDG2(1,IB) .EQ. KSS) IB2 = IB
  IF (IBNDG2(1,IB) .EQ. KSE) IB3 = IB
ENDDO
IBNDG2(1,IB2) = -9     ! Mark for delete
IBNDG2(3,IB1) = LCELL ! Reassign pointers
IBNDG2(2,IB3) = LCELL
```

This completes the grid fusion or collapse process. Reference can be made to subroutine G2CLPO in Appendix D for additional details. Flagged nodes are simply removed from the node-array tables and the pointers are realigned by generating new link-lists after each spatial adaptation cycle. This procedure is carried out in subroutine G2NODE. All of the pertinent details were shown here for both the division and collapse of cells to in order show the complexity of the logic for such procedures.

C.6 Extension of Spatially Resolved Region

As mentioned earlier the cells to be divided are stored in the array MRKDA2(IC), where IC varies from 1 to NCELD which indicates the total number of cells to be divided in a particular adaptation cycle. The width of the buffer zone is denoted by NXTDA2.

Suppose a typical cell, as shown in Figure (C.7), in the detected cluster is denoted by LC, which is pointed by IC; the neighbor cells are evaluated by the node-to-cell function for the nodes of LC. If any edge of LC is divided then the corresponding edge cell LS, LE, LN, or LW is already divided, and its subcells are at a higher spatial level than LC. Hence in this case it is not necessary to extend *through* that *edge*. For notational purposes these edge cells are shown to be at the same spatial level as the cell LC. Consider, for example, that the southern node exists (KS > 0) then we set the *extension pointer* KEP(5) of

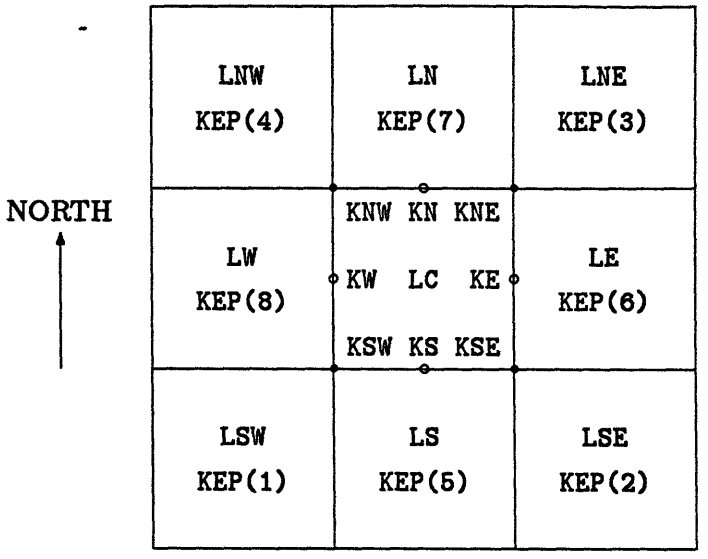

Figure C.7: Extension pointers associated with a given cell in MRKDA2 list.

this cell as zero indicating no extension through the southern edge. Otherwise, if the southern node does not exist KS = 0, then this extension pointer is set as

```
KEP(5) = LS = NEIBG2(2,KSW)
```

Note that the non-zero extension pointers are set equal to corresponding cells values. Similarly, the levels of the corner cells are examined and if the level of any of the corner cells is higher than that of LC then that corner extension pointer is set equal to zero. For example, for the south-west corner, the extension pointer is adjusted according as

```
IF ( LEVEL(LSW) .GT. LEVEL(LC) ) THEN
  KEP(1) = 0
ELSE
  KEP(1) = LSW = NEIBG2(1,KSW)
ENDIF
```

If at this point all the eight extension pointers are zeros, then there is no need to perform an extension for this particular cell and attention can be focused to examine

the next member of the detected cluster. However, if any pointer is non-zero then the corresponding cell must be checked for its presence in the MRKDA2 list, and if so, its pointer is set equal to zero. For example, if in the previous calculation it is observed that

```
KEP(6) = LE > 0
```

then examine all of the detected cluster [1]

```
MRKDA2(JC), JC = 1, NCELD
```

and if for some JC

```
MRKDA2(JC) = LE    ===>    KEP(6) = 0
```

If by now all the extension cell pointers are zero then proceed to examine the next cell in the detected cluster; otherwise collect all the non-zero extension pointers and store them in

```
MRKDA2(NCELD+IEXT)  for IEXT = 1, NEXTD
```

where NEXTD indicates the total number of cells in the current layer of the buffer zone.

The extended cells collected so far form the first layer of the buffer zone and subsequently only the cells in this layer must be examined for further extension if NXTDA2 exceeds unity. Furthermore the edges or corners of such cells should be appropriately *painted* to indicate that previous calculations of the extension procedure has already checked these edges and corners. For example, if KEP(5)=LS>0 in the current evaluation then the northeast and northwest corners as well as the northern edge must be painted

[1] Actually the search must take into account the newly added cells from the buffer zone in the MRKDA2 array, *i.e.*, use JC=1,NCELD+NEXTD for the search operation.

Cell	Indicator	Binary Form	Comment
LSW	1	0001	North-east corner painted
LSE	2	0010	North-west corner painted
LNE	4	0100	South-west corner painted
LNW	8	1000	South-east corner painted
LS	3	0011	Northern edge painted
LE	6	0110	Western edge painted
LN	12	1100	Southern edge painted
LW	9	1001	Eastern edge painted

Table C.1: Painting scheme for extension pointers.

since the calculation on LC has already examined its southern edge. The painting scheme for corner cells is shown by the first four rows in Table (C.1). The edge cell pointers are obtained by binary addition of the corresponding corner paints. The integer painting indicators PIND(1:MCELG2) are stored along with the boundary cells in the first layer at the interface of buffer and detected clusters. Note that for non-zero values of KEP(K), the painting indicators are given by

```
PIND (KEP(K)) = 2**(K-1)      for  K=1,2,3,4
PIND (KEP(K)) = 3*2**(K-1)    for  K=5,6,7
PIND (KEP(K)) = 9             for  K=8
```

If NXTDA2 is greater than unity, the cells in the first buffer layer are examined for

possible extension. Consider a typical cell LC, pointed by MRKDA2(IC) for IC between NCELD+1 and NCELD+NEXTD. The neighbor cells of LC are again examined and the extension pointers of these cells are set according as

1. If the edge node (KS, KE, KN, or KW) exists, set the corresponding KEP pointer equal to zero.
2. If the level of the corner cell (LSW, LSE, LNE, or LNW) is more than that of LC, set the corresponding KEP pointer equal to zero.
3. If the non-zero extension pointers exist in the MRKDA2 list then set them equal to zero. If all extension pointers are zero proceed to examine next cell in list.
4. If an edge or corner of LC is painted then set the KEP pointer of the corresponding cell equal to zero. This is explained further in the following. If all extension pointers are zero proceed to examine next cell in list.
5. Add all the non-zero extension pointers to the MRKDA2 list in positions NCELD+NEXTD+JX where JX varies from 1 to JEXTD.
6. Paint the appropriate edges and corners of the non-zero extension pointers in the current layer of buffer cells.
7. Proceed to examine the next IC in the loop.

The paints of the edges and corners of the cell under consideration is taken into account by the following sample code:

```
LC     = MRKDA2(IC) ! Find the actual cell in the loop
IPAINT = PIND(IC)   ! Find paint indicator

IF (IAND(IPAINT,X'00000001') .NE. 0) THEN    ! Northeast
    KEP(6) = 0      ! No extension through east
    KEP(3) = 0      ! No extension through northeast
    KEP(7) = 0      ! No extension through north
ENDIF
```

```
IF (IAND(IPAINT,X'00000002') .NE. 0) THEN    ! Northwest
    KEP(7) = 0      ! No extension through north
    KEP(4) = 0      ! No extension through northwest
    KEP(8) = 0      ! No extension through west
ENDIF
. . . . . .
```

After all cells are examined in the current layer of the buffer zone, the total number of cells in the clusters is adjusted as

```
NCELLD = NCELLD + NEXTD ! Total number of cells to be divided
NEXTD  = JEXTD          ! Number of cells in previous layer
JEXTD = 0               ! Number of cells in next layer
```

In this way, cells in the previous buffer layer are regarded as cells in a resolved cluster, and the current number of extended cells form the the next layer of the buffer zone whose neighbor cells will be next examined.

Once all of the extensions are completed, the cells in the **MRKCA2** list are examined and if any cell also appears in the overall buffer zone then it is removed from the fusion list. Subroutine **A2EXTD** in Appendix D contains additional details.

Appendix D

Program Listing

This appendix contains a listing of the STAR code that is based upon the spatio-temporal adaptive algorithm presented in this thesis. It also contains the listing of the block grid generator GNBLOC, some initial condition specification routines and GRAFIC interface routines. In addition, the appendix includes sample input files, synopsis of computer names, *etc.* The whole appendix appears in a separate volume that may be obtained by writing to:

Professor Judson R. Baron
Department of Aeronautics and Astronautics
Room 33-217, M.I.T.
Cambridge, Massachusetts 02139

Bibliography

[1] H. G. Adelman, J. L. Cambier, G. P. Menees, and J. A. Balboni. *Analytical and Experimental Validation of the Oblique Detonation Wave Engine Concept.* AIAA Paper 88-0097, January 1988.

[2] R. C. Aiken. *Stiff Computation.* Oxford University Press, New York, 1985.

[3] T. Aki. Computation of Unsteady Shock Wave Motion by the Modified Flux TVD Scheme. In F. G. Zhuang and Y. L. Zhu, editors, *Lecture Notes in Physics*, pages 86–90, Springer-Verlag, June 1986.

[4] T. Aki and F. Higashino. *A Numerical Study on Mach Reflection Around a Concave Surface.* Paper for seventh MRS, National Aerospace Laboratory, Chofu, Tokyo 182, Japan, June 1987.

[5] A. A. Amsden, J. D. Ramshaw, P. J. O'Rourke, and J. K. Dukowicz. *KIVA: A Computer Program for Two- and Three-Dimensional Fluid Flows with Chemical Reactions and Fuel Sprays.* LA-10245-MS, Los Alamos Scientific Laboratory, February 1985.

[6] G. Y. Anderson, D. P. Bencze, and B. W. Sanders. Ground Tests Confirm the Promise of Hypersonic Propulsion. *Aerospace America*, 25:38–42, September 1987.

[7] D. C. Arney and J. E. Flaherty. A Two-Dimensional Mesh Moving Technique for Time-Dependent Partial Differential Equations. *Journal of Computational Physics*, 67:124–144, 1986.

[8] E. H. Atta and J. Vadyak. A Grid Overlapping Scheme for Flowfield Computations About Multicomponent Configurations. *AIAA Journal*, 21(9):1271–1277, September 1983.

[9] T. J. Baker. *Developments and Trends in Three Dimensional Mesh Generation.* Transonic Symposium held at NASA Langley Research Center, Hampton, VA, April 1988.

[10] T. J. Baker, A. Jameson, and R. E. Vermeland. *Three-Dimensional Euler Solutions with Grid Embedding.* AIAA Paper 85-0121, January 1985.

[11] T. J. Barber and C. B. Cox. *Hypersonic Vehicle Propulsion: A CFD Application Case Study.* AIAA Paper 88-0475, January 1988.

[12] T. J. Bartel, G. F. Homicz, and M. A. Walker. *Comparisons of Monte-Carlo and PNS Calculations for Rarefied Flow Over Reentry Vehicle Configurations.* AIAA Paper 88-0465, January 1988.

[13] M. J. Berger. *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations.* STAN-CS-82-924, Department of Computer Science, Stanford University, August 1982.

[14] M. J. Berger. On Conservation at Grid Interfaces. *SIAM Journal of Numerical Analysis*, 24(5):967–984, October 1987.

[15] M. J. Berger and A. Jameson. Automatic Adaptive Grid Refinement for the Euler Equations. *AIAA Journal*, 23(4):561–568, April 1985.

[16] F. S. Billig. *Ramjet with Supersonic Combustion.* AGARD-NATO PEP Lecture Series No. 136, Ramjet and Ramrocket Propulsion Systems for Missiles, January 1986.

[17] C. W. Boppe. *Computational Transonic Flow about Realistic Aircraft Configurations.* NASA-CR-3243, May 1980.

[18] N. Botta, M. Pandolfi, and M. Germano. *Nonequilibrium Reacting Hypersonic Flow About Blunt Bodies: Numerical Prediction.* AIAA Paper 88-0514, January 1988.

[19] J. U. Brackbill and J. S. Saltzman. Adaptive Zoning for Singular Problems in Two Dimensions. *Journal of Computational Physics*, 46:342–368, 1982.

[20] K. N. C. Bray. *Nonequilibrium Flows*, chapter Chemical and Vibrational Nonequilibrium in Nozzle Flows, pages 59–157. Volume II, Marcel Dekker, Inc., 1970.

[21] E. G. Broadbent. Flows with Heat Addition. *Progress in Aerospace Science*, 17(2):93–108, 1976.

[22] T. D. Bui, A. K. Oppenheim, and D. T. Pratt. Recent Advances in Methods for Numerical Solution of ODE Initial Value Problems. *Journal of Computations and Applied Mathematics*, 11:283–296, 1984.

[23] T. R. A. Bussing. *A Finite Volume Method for the Navier Stokes Equations with Finite Rate Chemistry.* PhD thesis, M.I.T., August 1985.

[24] T. R. A. Bussing and E. M. Murman. *A Finite Volume Method for Calculation of Compressible Chemically Reacting Flows.* AIAA Paper 85-0331, CFDL-TR-85-3, January 1985.

[25] J. L. Cambier, H. G. Adelman, and G. P. Menees. *Numerical Simulations of an Oblique Detonation Wave Engine.* AIAA Paper 88-0063, January 1988.

[26] G. F. Carrier and C. E. Pearson. *Partial Differential Equations, Theory and Technique.* Academic Press, New York, 1976.

[27] S. R. Chakravarthy. *Euler Equations — Implicit Schemes and Implicit Boundary Conditions.* AIAA Paper 82-0228, January 1982.

[28] D. C. Chapman. *Philosophical Magazine.* 47(5):90, 1899.

[29] R. V. Chima and G. M. Johnson. Efficient Solution of the Euler and Navier-Stokes Equations with a Vectorized Multiple-Grid Algorithm. *AIAA Journal*, 23(1):23–32, January 1985.

[30] R. Courant and K. O. Friedrichs. *Supersonic Flows and Shock Waves.* Interscience Publishers, Inc., New York, 1948.

[31] C. F. Curtiss and J. O. Hirschfelder. Integration of Stiff Equations. *Proceedings of National Academy of Sciences*, 38:235–243, 1952.

[32] G. G. Dahlquist. A Special Stability Problem for Linear Multistep Methods. *BIT*, 3:27–43, 1963.

[33] J. F. Dannenhoffer III. *Grid Adaptation for Complex Two-dimensional Transonic Flows.* PhD thesis, M.I.T., CFDL-TR-87-10, August 1987.

[34] J. F. Dannenhoffer III and J. R. Baron. *Adaptive Procedure for Steady State Solution of Hyperbolic Equations.* AIAA Paper 84-0005, January 1984.

[35] J. F. Dannenhoffer III and J. R. Baron. *Grid Adaptation for the 2-D Euler Equations.* AIAA Paper 85-0484, January 1985.

[36] J. F. Dannenhoffer III and J. R. Baron. *Robust Grid Adaptation for Complex Transonic Flows.* AIAA Paper 86-0495, January 1986.

[37] S. M. Dash, N. Sinha, D. E. Wolf, B. J. York, W. J. Krawczyk, N. Rajendran, and T. B. Harris. *Computational Models for the Analysis/Design of Hypersonic Scramjet Components.* AIAA Papers 86-1595 and 86-1596, June 1986.

[38] S. M. Dash, D. E. Wolf, and N. Sinha. Parabolized Navier-Stokes Analysis of Three-Dimensional Supersonic and Subsonic Jet Mixing Problems. *AIAA Journal,* 24(8):1252–1253, August 1986.

[39] R. L. Davis. *The Prediction of Compressible, Laminar Viscous Flows Using a Time-Marching Control Volume and Multigrid Technique.* AIAA Paper 83-1896, July 1983.

[40] S. R. de Groot and P. Mazur. *Non-equilibrium Thermodynamics.* Dover Publications, Inc., 1984.

[41] J. P. Drummond. *Numerical Investigation of the Perpendicular Injector Flow Field in a Hydrogen Fueled Scramjet.* AIAA Paper 79-1482, July 1979.

[42] J. P. Drummond, M. Y. Hussaini, and T. A. Zang. Spectral Methods for Modelling Supersonic Chemically Reacting Flow Fields. *AIAA Journal,* 24(9):1461–1467, September 1986.

[43] J. P. Drummond, R. C. Rogers, and M. Y. Hussaini. *A Detailed Numerical Model of a Supersonic Reacting Mixing Layer.* AIAA Paper 86-1427, 1986.

[44] J. P. Drummond and E. Weidner. *Numerical Study of Scramjet Engine Flowfield.* AIAA Paper 81-0186R, 1981.

[45] G. L. Dugger, F. S. Billig, and W. H. Avery. *Hypersonic Propulsion Studies at Applied Physics Laboratory.* Presented at Ramjet Technical Session of Joint Institute of Aerospace Sciences and American Rocket Society, Summer Meeting, John Hopkins University, June 1961.

[46] D. L. Dwoyer and A. Kumar. *Computational Analysis of Hypersonic Airbreathing Aircraft Flow Fields.* AIAA Paper 87-0279, January 1987.

[47] H. A. Dwyer, R. J. Kee, and B. R. Sanders. Adaptive Grid Method for Problems in Fluid Mechanics and Heat Transfer. *AIAA Journal*, 18(10):1205–1212, October 1980.

[48] S. Eberhardt and K. Brown. *A Shock Capturing Technique for Hypersonic, Chemically Relaxing Flows.* AIAA Paper 86-0231, January 1986.

[49] P. R. Eiseman. Alternating Direction Adaptive Grid Generator. *AIAA Journal*, 23(4):551–560, April 1985.

[50] W. H. Enright and T. E. Hull. *Numerical Methods for Differential Systems*, chapter Comparing Numerical Methods for the Solution of Stiff System of ODEs Arising in Chemistry, pages 45–66. Academic Press, New York, 1976.

[51] A. Ferri. *High Speed Aerodynamics and Jet Propulsion.* Volume 6, Princeton University Press, Princeton, New Jersey, 1954.

[52] N. C. Freeman. Non-equilibrium Flow of an Ideal Dissociating Gas. *Journal of Fluid Mechanics*, 4:407–425, August 1958.

[53] K. O. Friedrichs. *High Speed Aerodynamics and Jet Propulsion.* Volume 6, Princeton University Press, Princeton, New Jersey, 1954.

[54] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[55] R. J. Gelinas, S. K. Doss, and K. Miller. The Moving Finite Element Method: Applications to General Partial Differential Equations with Multiple Large Gradients. *Journal of Computational Physics*, 40:202–249, 1981.

[56] T. P. Gielda, L. G. Hunter, and J. R. Chawner. *Efficient Parabolized Navier-Stokes Solutions of Three-Dimensional, Chemically Reacting Scramjet Flowfields.* AIAA Paper 88-0096, January 1988.

[57] M. Giles. *UNSFLO: A Numerical Method for Unsteady Inviscid Flow in Turbomachinery.* CFDL-TR-86-6, M.I.T., December 1986.

[58] P. A. Gnoffo. A Finite-Volume, Adaptive Grid Algorithm Applied to Planetary Entry Flowfields. *AIAA Journal*, 21(9):1249–1254, September 1983.

[59] M. D. Griffin, F. S. Billig, and M. E. White. *Applications of Computational Techniques in the Design of Ramjet Engines.* AIAA Paper 8-7026, 1983.

[60] M. G. Hall. *Cell-Vertex Multigrid Schemes for Solution of the Euler Equations.* Technical Report 2029, Royal Aircraft Establishment, March 1985.

[61] M. G. Hall and M. D. Salas. *Comparison of Two Multigrid Methods for the Two-dimensional Euler Equations.* AIAA Paper 85-1515-CP, July 1985.

[62] J. R. Henry and G. Y. Anderson. *Design Considerations for the Airframe Integrated Scramjet.* NASA-TM-X-2895, 1973.

[63] K. A. Hessenius and T. H. Pulliam. *A Zonal Approach to Solution of the Euler Equations.* AIAA Paper 82-0969, June 1982.

[64] J. E. Holcomb. *Development of a Grid Generator to Support 3-D Multizone Navier-Stokes Analysis.* AIAA Paper 87-0203, January 1987.

[65] J. E. Holcomb and R. G. Hindman. *Development of a Dynamically Adaptive Grid Method for Multidimensional Problems.* AIAA Paper 84-1668, June 1984.

[66] T. L. Holst and D. Brown. Transonic Airfoil Calculations Using Solution-Adaptive Grids. *AIAA Journal*, 21(2):304–306, February 1983.

[67] C. M. Hung and T. J. Barth. *Computation of Hypersonic Flow Through a Narrow Expansion Slot.* AIAA Paper 88-0232, January 1988.

[68] A. Jameson, W. Schmidt, and E. Turkel. *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes.* AIAA Paper 81-1259, June 1981.

[69] P. J. Johnston, A. H. Whitehead, and G. T. Chapman. Fitting Aerodynamics and Propulsion into the Puzzle. *Aerospace America*, 25:32–37, September 1987.

[70] E. Jouguet. La Theorie Thermodynamique de la Propagation des Explosions. *Proceedings of the Second International Congress for Applied Mechanics, Zurich*, pp. 12–22, September 1926. Also see *Journal of Mathematics*, pp. 347, 1905, and pp. 6, 1906; *Mechanique de Explosifs*, O. Dien et Fils, Paris, 1907.

[71] J. G. Kallinderis and J. R. Baron. *Adaptation Methods for a New Navier-Stokes Algorithm.* AIAA Paper 87-1167-CP, 1987.

[72] R. J. Kee and H. A. Dwyer. Review of Stiffness and Implicit Finite-Difference Methods in Combustion Modeling. *Progress in Astronautics and Aeronautics,* 76:485–500, August 1979.

[73] J. Krispin and J. D. Anderson. *Accelerating the Computation of Steady State Solutions for Nonequilibrium, Chemically Reacting Flow Fields.* AIAA Paper 88-0513, January 1988.

[74] A. Kumar. *Numerical Analysis of the Scramjet-Inlet Flow Field by Using Two-Dimensional Navier-Stokes Equations.* AIAA Paper 81-0185, January 1981.

[75] A. Kumar, D. M. Bushnell, and M. Y. Hussaini. *A Mixing Augmentation Technique for Hypervelocity Scramjets.* AIAA Paper 87-1882, June 1987.

[76] P. D. Lax and B. Wendroff. System of Conservation Laws. *Communications in Pure and Applied Mathematics,* 13:217–327, 1960.

[77] H. W. Liepmann and A. Roshko. *Elements of Gas Dynamics.* John Wiley and Sons, Inc., 1957.

[78] M. J. Lighthill. Dynamics of a Dissociating Gas. *Journal of Fluid Mechanics,* 2:1–32, January 1957.

[79] R. Löhner. Some Useful Data Structures for the Generation of Unstructured Grids. *Communications in Applied Numerical Methods,* 4(1):123–135, January 1988.

[80] R. Löhner, K. Morgan, J. Peraire, and M. Vahdati. *Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier-Stokes Equations.* ICASE Report Number 87-4, NASA-CR-178233, January 1987.

[81] R. Löhner, K. Morgan, J. Peraire, and O. C. Zienkiewicz. *Finite Element Methods for High Speed Flows.* AIAA Paper 85-1531, July 1985.

[82] R. Löhner, K. Morgan, and O. C. Zienkiewicz. The Use of Domain Splitting With an Explicit Hyperbolic Solver. *Computer Methods in Applied Mechanics and Engineering,* 45(1-3):313–329, September 1984.

[83] J. A. Lordi, D. W. Boyer, M. G. Dunn, K. K. Smolarek, and C. E. Wittliff. *Description of Non-equilibrium Effects on Simulations of Flows About Hypersonic Vehicles.* AIAA Paper 88-0476, January 1988.

[84] M. Mani, S. N. Tiwari, and J. P. Drummond. *Investigation of Two-dimensional Chemically Reacting and Radiating Supersonic Channel Flows.* AIAA Paper 88-0462, January 1988.

[85] M. Mani, S. N. Tiwari, and J. P. Drummond. *Numerical Solutions of Chemically Reacting and Radiating Flows.* AIAA Paper 87-0324, January 1987.

[86] M. Martinez-Sanchez. Private Communication, 1988.

[87] D. Mavriplis and A. Jameson. *Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Meshes.* AIAA Paper 87-0353, 1987.

[88] R. May and J. Noye. *Computational Techniques for Differential Equations*, chapter The Numerical Solutions of Ordinary Differential Equations: Initial Value Problems, pages 1–94. North-Holland Inc., New York, 1984.

[89] P. A. McMurtry, W. Jou, J. J. Riley, and R. W. Metcalfe. *Direct Numerical Simulations of a Reacting Mixing Layer with Chemical Heat Release.* AIAA Paper 85-0143, January 1985.

[90] R. E. Meyer. *Mordern Developments in Fluid Dynamics, High Speed Flow.* Volume 1, Oxford University Press, New York, 1953.

[91] K. Miller and R. N. Miller. Moving Finite Elements. *SIAM Journal of Numerical Analysis*, 18(6):1019–1032, December 1981.

[92] R. B. Morrison. *Evaluation of the Oblique Detonation Wave Ramjet.* NASA-CR-145358, January 1978.

[93] R. B. Morrison. *Oblique Detonation Wave Ramjet.* NASA-CR-159192, January 1980.

[94] J. N. Moss, G. A. Bird, and V. K. Dogra. *Nonequilibrium Thermal Radiation for an Aeroassist Flight Experiment Vehicle.* AIAA Paper 88-0081, January 1988.

[95] K. Nakahashi and G. S. Deiwert. Self-Adaptive-Grid Method with Application to Airfoil-Flow. *AIAA Journal*, 25(4):513–520, April 1987.

[96] R. Ni. A Multiple Grid Scheme for Solving the Euler Equations. *AIAA Journal*, 20(11):1565–1571, November 1982.

[97] R. W. Noack and D. A. Anderson. *Solution Adaptive Grid Generation Using Parabolic Partial Differential Equations.* AIAA Paper 88-0315, January 1988.

[98] G. B. Northam and G. Y. Anderson. *Supersonic Combustion Ramjet Research at Langley.* AIAA Paper 86-0159, January 1986.

[99] J. T. Oden, T. Strouboulis, and P. Devloo. *An Adaptive Finite Element Strategy for Complex Flow Problems.* AIAA Paper 87-0557, January 1985.

[100] E. S. Oran and J. P. Boris. *Numerical Simulation of Reactive Flow.* Elsevier Science Publishing Company, Inc., 1987.

[101] S. Osher and R. Sanders. Numerical Approximations to Nonlinear Conservation Laws With Locally Varying Time and Space Grids. *Mathematics of Computations*, 41(164):321–336, October 1983.

[102] D. Papamoschou and A. Roshko. *Observations of Supersonic Free Shear Layers.* AIAA Paper 86-0162, January 1986.

[103] C. Park. *Two-Temperature Interpretation of Dissociation Rate Data for N_2 and O_2.* AIAA Paper 88-0458, 1988.

[104] J. Peraire, K. Morgan, J. Peiro, and O. C. Zienkiewicz. *An Adaptive Finite Element Method for High Speed Flows.* AIAA Paper 87-0558, January 1987.

[105] M. M. Pervaiz and J. R. Baron. *Spatio-temporal Adaptation Algorithm for Two-Dimensional Reacting Flow.* AIAA Paper 88-0510, January 1988.

[106] M. M. Pervaiz and J. R. Baron. Temporal and Spatial Adaptive Algorithm for Reacting Flow. *Communications in Applied Numerical Methods*, 4(1):97–111, January 1988.

[107] R. Peyret and T. Taylor. *Computational Methods for Fluid Flow.* Springer-Verlag, Inc., 1983.

[108] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes.* Cambridge University Press, 1986.

[109] K. Radhakrishnan. Integrating Combustion Kinetic Rate Equations by Selective Use of Stiff and Nonstiff Methods. *AIAA Journal*, 25(11):1449–1455, November 1987.

[110] M. M. Rai. A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations. *Journal of Computational Physics*, 62:472–503, 1986.

[111] M. M. Rai. *Navier-Stokes Simulations of Rotor-Stator Interaction Using Patched and Overlaid Grids.* AIAA Paper 85-1519, 1985.

[112] J. D. Ramshaw. Partial Chemical Equilibrium in Fluid Dynamics. *Physics of Fluids*, 23(4):675–680, April 1980.

[113] C. L. Reed and S. L. Karman. *Multiple-block Grid Method Applied to Complex 3-D Geometries.* Presented at Society for Industrial and Applied Mathematics, Boston, July 1986.

[114] W. C. Rivard, O. A. Farmer, and T. D. Butler. *RICE: A Computer Program for Multicomponent Chemically Reactive Flows at All Speeds.* LA-5812, Los Alamos Scientific Laboratory, March 1975.

[115] R. C. Rogers and W. Chinitz. Using a Global Hydrogen-Air Combustion Model in Turbulent Reacting Flow Calculations. *AIAA Journal*, 21(4):586–592, April 1983.

[116] R. Sedney. *Nonequilibrium Flows*, chapter The Method of Characteristics, pages 159–225. Volume II, Marcel Dekker, Inc., 1970.

[117] J. S. Shang and E. Josyula. *Numerical Simulations of Non-Equilibrium Hypersonic Flow Past Blunt Bodies.* AIAA Paper 88-0512, January 1988.

[118] A. H. Shapiro. *The Dynamics and Thermodynamices of Compressible Flow.* Volume 2, Ronald, New York, 1954.

[119] R. A. Shapiro and E. M. Murman. *Cartesian Grid Finite Element Solutions to the Euler Equations.* AIAA Paper 87-0559, 1987.

[120] Y. Sheng and J. P. Sislian. *A Model of a Hypersonic Two-Dimensional Oblique Detonation Wave Ramjet.* UTIAS Technical Note No. 257, CN ISSN 0082-5263, Institute of Aerospace Studies, University of Toronto, July 1985.

[121] M. D. Smooke and M. L. Koszykowski. Fully Adaptive Solutions of One-Dimensional Mixed Initial-Boundary Value Problems with Applications to Unstable Problems in Combustion. *SIAM Journal of Scientific and Statistical Computing*, 7(1):301–321, January 1986.

[122] J. F. Stalnaker, M. A. Robinson, L. W. Spradley, S. C. Kurzius, and J. Thoenes. *Development of the General Interpolants Method for the CYBER 200 Series of Supercomputers.* Technical Report LMSC-HREC TR D867354, Lockheed Missiles and Space Company, October 1983.

[123] J. Steinhoff. Blending Methods for Grid Generation. *Journal of Computational Physics*, 65:370–385, 1986.

[124] R. A. Suehla. *Estimated Viscosities and Thermal Conductivities of Gases at High Temperatures.* NASA-TR-R-132, 1962.

[125] B. K. Swartz and B. Wendroff. *AZTEC: A Front Tracking Code Based on Godunov's Method.* ICASE Report Number 86-18, NASA-CR-178076, March 1986.

[126] Y. Takano. An Application of the Random Choice Method to Reactive Gas with Many Chemical Species. *Journal of Computational Physics*, 67:173–187, 1986.

[127] S. Taki and T. Fujiwara. *Dynamics of Shock Waves, Explosions and Detonations*, chapter Numerical Simulations on the Establishment of Gaseous Detonation, pages 186–200. Volume 94, Progress in Aeronautics and Astronautics, 1985.

[128] J. W. Thomas, M. Heroux, S. McKay, S. McCormick, and A. Thomas. Applications of the Fast Adaptive Composite Grid Method to Computation Fluid Dynamics. In C. Taylor, W. G. Habashi, and M. Hafez, editors, *Numerical Methods in Laminar and Turbulent Flows*, pages 1071–1082, Pineridge Press, 1987.

[129] J. F. Thompson. Grid Generation Techniques in Computational Fluid Dynamics. *AIAA Journal*, 22(11):1505–1523, November 1984.

[130] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation.* Elsevier Science Publishing Company, Inc., 1985.

[131] T. Toong. *Combustion Dynamics.* McGraw-Hill Book Company, 1983.

[132] E. B. Turner and G. Emanuel. *The NEST Chemistry Computer Program.* Air Force report number SAMSO-TR-70-311, Vol. I, July 1970.

[133] W. J. Usab. *Embedded Mesh Solutions of the Euler Equation Using a Multiple-grid Method.* PhD thesis, M.I.T., CFDL-TR-84-2, December 1983.

[134] G. Van Wylen and R. Sonntag. *Fundamentals of Classical Thermodynamics.* John Wiley and Sons, Inc., 1978.

[135] I. M. Vardavas. Modelling Reactive Gas Flows within Shock Tunnels. *Austrailian Journal of Physics*, 37:157–177, 1984.

[136] G. Vincenti and C. Kruger. *Introduction to Physical Gas Dynamics.* John Wiley and Sons, Inc., 1965.

[137] J. C. Wang and G. F. Widhopf. *Numerical Simulation of Blast Flowfields Using a High Resolution TVD Finite Volume Scheme.* AIAA Paper 87-1320, June 1987.

[138] F. M. White. *Viscous Fluid Flow.* McGraw-Hill Book Company, 1974.

[139] M. E. White, J. P. Drummond, and A. Kumar. Evolution and Application of CFD Techniques for Scramjet Engine Analysis. *Journal of Propulsion*, 3:423–439, September 1987.

[140] C. R. Wilke. A Viscosity Equation for Gas Mixtures. *Journal of Chemical Physics*, 18:517–519, April 1950.

[141] F. Williams. *Combustion Theory.* Addison-Wesley, 1965.

[142] D. E. Wolf, R. A. Lee, and S. M. Dash. *Parabolized Navier-Stokes Analysis of Scramjet Hypersonic Nozzle Flowfields.* AIAA Paper 87-1897, June 1987.

[143] P. Woodward and P. Collella. The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks. *Journal of Computational Physics*, 54:115–173, 1984.

[144] J. Y. Yang, C. K. Lombard, and D. Bershader. Numerical Simulation of Transient Inviscid Shock Tube Flows. *AIAA Journal*, 25(2):245–251, February 1987.

[145] H. C. Yee. *Upwind and Symmetric Shock-Capturing Schemes.* NASA Technical Memorandum 89464, May 1987.

[146] B. J. York, N. Sinha, and S. M. Dash. *Computational Models for Chemically-Reacting Hypersonic Flow.* AIAA Paper 88-0509, January 1988.

[147] V. Y. Young and H. C. Yee. *Numerical Simulation of Shock Wave Diffraction by TVD Schemes.* AIAA Paper 87-0112, January 1987.

Appendix D

Computer Code

This appendix is the source of information for executing the STAR code that is based upon the spatio-temporal adaptive algorithm presented in the first volume. It also contains the listing of the block grid generator GNBLOC, some initial condition and initial coarse grid generators and GRAFIC interface routines. Only the listings of the programs for two spatial dimensions are provided here. The subroutines from the graphics package GRAFIC are also not included. In addition sample input files and synopsis of computer names for each module is provided.

The initial coarse grid and initial conditions on this grid are generated by running separate programs. These programs generate the files `INPUTG.DAT` for grid and `INPUTD.DAT` for initial dependent variables at each computational node. The chemistry deck `INPUTC.DAT` includes information such as stoichiometric coefficients, specific heats, *etc.* If special procedures are needed in addition to the normal integration process then the deck `INPUTS.DAT` is needed that includes a lists of commands to be executed. An example of special procedures is pre-embedding of an initial coarse grid. For a complete list of special procedures the reader is referred to subroutine `E2SCHO` in Section D.3. The file `INPUTI.DAT` contains a list of parameters for the current run of the STAR code.

Once all the input is read and all special procedures are performed the program starts integrating on a cell by cell basis. The integration continues until a specified time-station or for a fixed number of time-strides. For steady state calculations the integration proceeds to a desired level of convergence. The program then writes the dump output on the file `JPNTWR.DAT` that includes information on the whole data structure and most common block variables. With this file the graphical output can be examined and

plots generated. This file can also be used to restart the calculations at a later time with a possibility of a different set of parameters in the subsequent simulation. These parameters are also read from a new file INPUTI.DAT for the restart case. Figure (D.1) shows the organisation of the overall computer program.

The appendix is divided into six sections. The first section lists the programs that generate the coarse structured grids and initial conditions for simple geometries. These include the programs to generate data for the moving shock over a circular arc bump and inside a bent duct. The initial condition generators include programs for shock tube and moving shock waves. The second section contains the listing for an interactive block grid generator that can be used to generate meshes for more complicated geometries. The third section pertains to the STAR code, whereas the next two sections list the utility and GRAFIC interface routines. The last section contains two sample input files for the STAR code.

D.1 Initial coarse grid and initial conditions

In order to keep the integration procedure independent of the geometry of the individual problems and the specific initial distribution of state vectors, the STAR code requires the allocation of grid points and initial conditions through separate programs. These programs generate the files INPUTG.DAT for grid and INPUTD.DAT for initial dependent variables at each computational node in formats that STAR code understands. The programs that generate these files can be a part of the same routine or could be generated separately. For some problems initial condition generator is not needed explicitly since uniform inflow conditions are specified as a starting condition. The file INPUTG.DAT writes the x and y coordinates at each computational node and the boundary arrays at the boundary points. The file INPUTD.DAT writes all dependent variables at the same computational nodes.

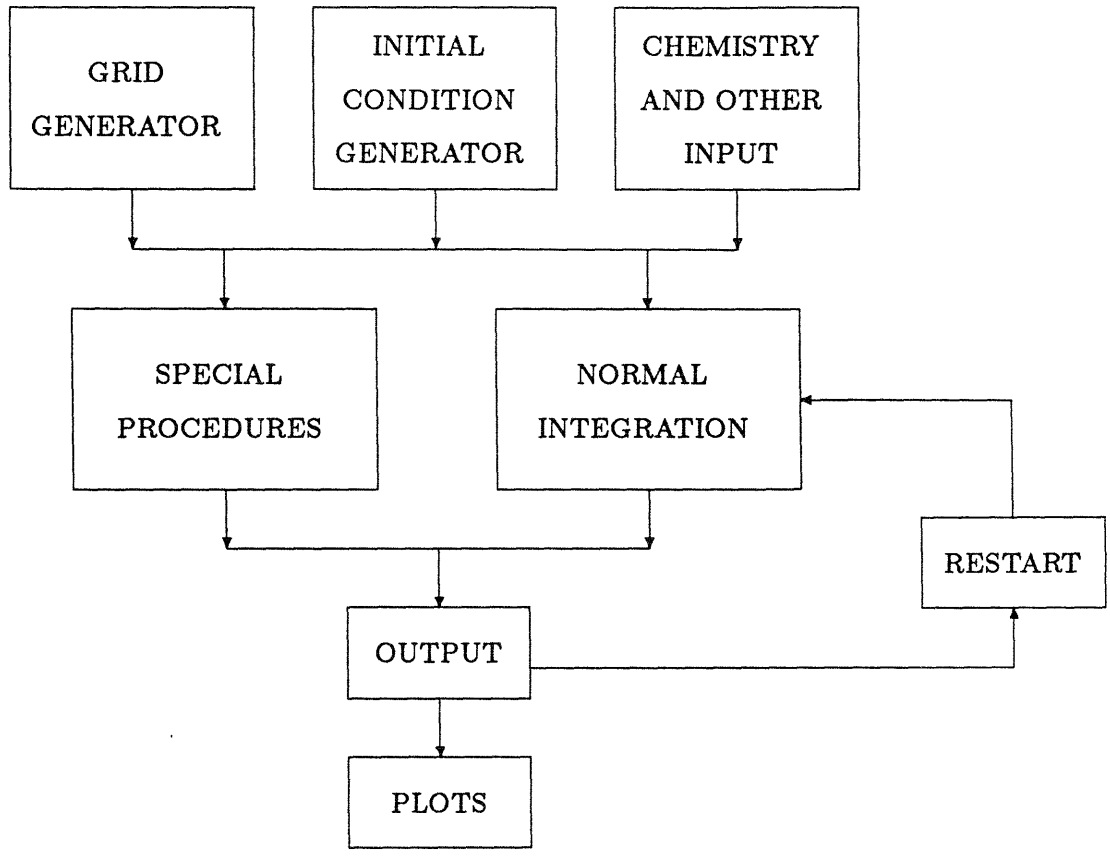

Figure D.1: Organization of the computer program.

D.1.1 Initial conditions

The programs for three initial condition types are listed here. These are for conditions across a contact surface of a shock tube, a moving shock wave involving a step function for frozen or equilibrium flows, and a non-equilibrium moving shock.

Shock tube

The program MOCONC calculates the initial conditions across the contact surface for a shock tube. The gas in the two sections is assumed to be a perfect gas or a Lighthill gas. For the Lighthill gas the degree of dissociation in the two sections can be specified arbitrarily for frozen flow, whereas for equilibrium flow these are calculated from an algebraic expression.

```
      PROGRAM MOCONC
C*********************************************************************
C
C     THIS PROGRAM GENERATES THE FROZEN OR EQUILIBRIUM CONDITIONS FOR
C     A SHOCK TUBE WITH A CONTACT SURFACE.  CONDITIONS AT THE INLET
C     AND EXIT ARE ASSUMED TO BE KNOWN AND READ FROM FILE MOCONC.DAT.
C
C     SUBROUTINES CALLED: EQUCAL      UTILITY ROUTINE
C
C*********************************************************************

      INCONC = 51
      JPRINT = 6
      IOSHOC = 8
      IOPIPE = 9
C
C     OPEN THE APPROPRAITE UNITS
C
      OPEN (UNIT = INCONC, FILE = 'MOCONC.DAT', STATUS = 'OLD')
      OPEN (UNIT = IOSHOC, FILE = 'MOCONC.OUT', STATUS = 'NEW')
      OPEN (UNIT = IOPIPE, FILE = 'MOCONC.IOT', STATUS = 'NEW')
C
C     INPUT THE FOLLOWING QUANTITIES FROM FILE MOCONC.DAT
C
C        ALPHAI    INITIAL ALPHA AT INLET
C        ALPHAE    INITIAL ALPHA AT EXIT
C        RHOI      DENSITY AT INLET IN kg/m**3
C        TEMPI     TEMPERATURE AT INLET IN Kelvins
C        RHOE      DENSITY AT OUTLET IN kg/m**3
C        TEMPE     TEMPERATURE AT OUTLET IN Kelvins
C        SHKMAC    MACH NUMBER OF THE SHOCK
```

```
C          PHI       REACTION PARAMETER (<0 MEANS EQUILIBRIUM)
C          IALP      INLET/EXIT SELECTOR FOR REFERENCE QUANTITIES
C                     EXIT:0, INLET:1
C
      READ (INCONC,*) ALPHAI
      READ (INCONC,*) ALPHAE
      READ (INCONC,*) RHOI
      READ (INCONC,*) TEMPI
      READ (INCONC,*) RHOE
      READ (INCONC,*) TEMPE
      READ (INCONC,*) SHKMAC
      READ (INCONC,*) PHI
      READ (INCONC,*) IALP
C
C     WRITE DOWN THESE VALUES IN MOCONC.OUT FOR THE PURPOSE OF KEEPING
C     A RECORD
C
      WRITE(IOSHOC,10) ALPHAI,ALPHAE,RHOI,TEMPI,RHOE,TEMPE,SHKMAC,PHI
10    FORMAT (' INITIAL ALPHA AT INLET             =',G14.5/
     1        ' INITIAL ALPHA AT EXIT              =',G14.5/
     1        ' RHO INLET                          =',G14.5/
     1        ' TEMPERAURE INLET                   =',G14.5/
     1        ' RHO OUTLET                         =',G14.5/
     1        ' TEMPERAURE OUTLET                  =',G14.5/
     1        ' SHOCK MACH NUMBER                  =',G14.5/
     1        ' PHI -- REACTION PARAMETER          =',G14.5/)
C
C     NOW COMPUTE THE DEPENDENT VARIABLES, VALUES FOR OXYGEN ARE ASSUMED
C     HERE FOR THE LIGHTHILL MODEL
C
      THETAD = 59500.
      RHOD   = 150.E03
      UGASFL = 8314.3
      AMWTA  = 16.0
      RGAS   = 0.5*UGASFL/AMWTA
C
      IF (PHI .GE. 0.) GOTO 100

C     COMPUTE THE EQUILIBRIUM VALUES AT THE INLET AND EXIT

      CALL EQUCAL(TEMPI,THETAD,RHOI,RHOD,ETRAT,RHORAT,ALPHAI)
      CALL EQUCAL(TEMPE,THETAD,RHOE,RHOD,ETRAT,RHORAT,ALPHAE)

100   PRESSI = RHOI*(1.+ALPHAI)*RGAS*TEMPI
      PRESSE = RHOE*(1.+ALPHAE)*RGAS*TEMPE

      IF (IALP .NE. 0) THEN
C        USE TEMPI AND RHOI AS INLET VALUES FOR REFERENCE CONDITIONS
         RHORFL = RHOI
         PRESFL = PRESSI
         ALPHAR = ALPHAI
      ELSE
C        USE TEMPE AND RHOE AS EXIT VALUES FOR REFERENCE CONDITIONS
         RHORFL = RHOE
         PRESFL = PRESSE
         ALPHAR = ALPHAE
      ENDIF
```

```
C
C       COMPUTE THE NORMALIZED QUANTITIES

        ONEPA1 = 1. + ALPHAR
C       ONEPA1 IS 1 + ALPHAR
C       COMPUTE REFERENCE TEMPERATURE
        TREFFL = PRESFL/(ONEPA1*RHORFL*RGAS)
C       COMPUTE REFERENCE VELOCITY
        UREFFL = SQRT(PRESFL/RHORFL)
        RHOINL = RHOI
        TINLET = TEMPI
        PINLET = PRESSI
        RHOEXT = RHOE
        PEXIT  = PRESSE
        TEXIT  = TEMPE
C
C       NORMALIZE ALL VALUES
C
        RHOI   = RHOI/RHORFL
        PRESSI = PRESSI/PRESFL
        TEMPI  = TEMPI/TREFFL
        RHOE   = RHOE/RHORFL
        PRESSE = PRESSE/PRESFL
        TEMPE  = TEMPE/TREFFL
        THETAD = THETAD/TREFFL
        RHOD   = RHOD/RHORFL
C
C       WRITE DOWN THE NORMALIZED VALUES IN MOCONC.OUT FOR RECORD.
C
        WRITE(IOSHOC,210) RHOE,PRESSE,TEMPE, RHOI,PRESSI,TEMPI,
     1                    RHORFL,PRESFL,TREFFL, RHOINL,PINLET,TINLET,
     1                    RHOEXT,PEXIT,TEXIT,ALPHAI,ALPHAE,RHOD,THETAD,
     1                    UREFFL,RGAS
210        FORMAT (' RHOE  =',G14.5,5X,'PRESSE=',G14.5,5X,'TEMPE =',G14.5/
     1             ' RHOI  =',G14.5,5X,'PRESSI=',G14.5,5X,'TEMPI =',G14.5/
     2             ' RHORFL=',G14.5,5X,'PRESFL=',G14.5,5X,'TREFFL=',G14.5/
     3             ' RHOINL=',G14.5,5X,'PINLET=',G14.5,5X,'TEMPI =',G14.5/
     3             ' RHOEXT=',G14.5,5X,'PEXIT =',G14.5,5X,'TEXIT =',G14.5/
     4             ' ALPHAI=',G14.5,5X,'ALPHAE=',G14.5,5X,'RHOD  =',G14.5/
     5             ' THETAD=',G14.5,5X,'UREFFL=',G14.5,5X,'RGAS  =',G14.5/)
C
C       WRITE DOWN THE VALUES IN MOCONC.IOT SO THAT THESE CAN BE READ LATER
C       BY A GRID GENERATOR PROGRAM LIKE BEPIPE.FOR
C
        WRITE(IOPIPE,220) ALPHAI
        WRITE(IOPIPE,220) ALPHAE
        WRITE(IOPIPE,220) ALPHAR
        WRITE(IOPIPE,220) RHOI
        WRITE(IOPIPE,220) RHOE
        WRITE(IOPIPE,220) PRESSI
        WRITE(IOPIPE,220) PRESSE
        WRITE(IOPIPE,220) TEMPI
        WRITE(IOPIPE,220) TEMPE
        WRITE(IOPIPE,220) TREFFL
        WRITE(IOPIPE,220) RHORFL
        WRITE(IOPIPE,220) SHKMAC
220     FORMAT (G14.5)
```

```
C       EXAMPLE FILE : MOCONC.DAT
C 0.20                 ALPHAI
C 0.20                 ALPHAE
C 8.75939              RHOI
C 3343.2               TINLET
C 0.25983              RHOE
C 3343.2               TOUTLET
C 2.                   SHKMAC
C 0.01                 PHI
C 0                    IALP

        END
```

Moving shock for frozen or equilibrium flow

The program MOSHOC calculates the initial conditions across a moving shock. The gas in the two sections is assumed to be a perfect gas or a Lighthill gas. For the Lighthill gas the degree of dissociation in the two sections can be specified arbitrarily for frozen flow, whereas for equilibrium flow these are calculated from an algebraic expression.

```
        PROGRAM MOSHOC

C*********************************************************************
C
C       THIS PROGRAM GENERATES THE FROZEN OR EQUILIBRIUM CONDITIONS FOR
c       A MOVING SHOCK.  CONDITIONS AT THE INLET (BEHIND SHOCK) ARE
C       ASSUMED TO BE KNOWN AND READ FROM FILE MOSHOC.DAT.
C
C       SUBROUTINES CALLED: EQUCAL      UTILITY ROUTINE
C
C*********************************************************************

        INSHOC = 51
        JPRINT = 6
        IOSHOC = 8
        IOPIPE = 9
C
C       VALUES FOR OXYGEN ARE ASSUMED HERE FOR THE LIGHTHILL MODEL
C
        THETAD = 59500.
        RHOD   = 150.E03
        UGASFL = 8314.3
        AMWTA  = 16.0
        RGAS   = 0.5*UGASFL/AMWTA
C
C       OPEN THE APPROPRAITE UNITS
C
        OPEN (UNIT = INSHOC, FILE = 'MOSHOC.DAT', STATUS = 'OLD')
        OPEN (UNIT = IOSHOC, FILE = 'MOSHOC.OUT', STATUS = 'NEW')
```

```
      OPEN (UNIT = IOPIPE, FILE = 'MOSHOC.IOT', STATUS = 'NEW')
C
C     INPUT THE FOLLOWING QUANTITIES FROM FILE MOSHOC.DAT
C
C        ALPHAI    INITIAL ALPHA AT INLET
C        ALPHAE    INITIAL ALPHA AT EXIT
C        RHOI      DENSITY AT INLET IN kg/m**3
C        TEMPI     TEMPERATURE AT INLET IN Kelvins
C        SHKMAC    MACH NUMBER OF THE SHOCK
C        REBRI     DENSITY RATIO (GUESS)
C        TEBTI     TEMPERATURE RATIO (GUESS)
C        PHI       REACTION PARAMETER (<0 MEANS EQUILIBRIUM)
C        IALP      WEIGHT FACTOR FOR EQUILIBRIUM ALPHA, NORMAL VALUE=0
C                   AVERAGE VALUE=1, FOR VERY SLOW CONVERGENCE VALUE=10
C              OR   INLET/EXIT SELECTOR FOR REFERENCE QUANTITIES FOR
C                   THE FROZEN FLOW; EXIT:0, INLET:1
C        IGAS      PARAMETER INDICATING THE TYPE OF GAS USED
C                   0: LIGHTHILL         1: PERFECT
C
C
      READ (INSHOC,*) ALPHAI
      READ (INSHOC,*) ALPHAE
      READ (INSHOC,*) RHOI
      READ (INSHOC,*) TEMPI
      READ (INSHOC,*) SHKMAC
      READ (INSHOC,*) REBRI
      READ (INSHOC,*) TEBTI
      READ (INSHOC,*) PHI
      READ (INSHOC,*) IALP
      READ (INSHOC,*) IGAS
C
C     WRITE DOWN THESE VALUES IN MOSHOC.OUT FOR THE PURPOSE OF KEEPING
C     A RECORD
C
      WRITE(IOSHOC,10) ALPHAI,ALPHAE,RHOI,TEMPI,SHKMAC,REBRI,TEBTI,PHI
10    FORMAT (' INITIAL ALPHA AT INLET            =',G14.5/
     1        ' INITIAL ALPHA AT EXIT             =',G14.5/
     1        ' RHO INLET                         =',G14.5/
     1        ' TEMPERAURE INLET                  =',G14.5/
     1        ' SHOCK MACH NUMBER                 =',G14.5/
     1        ' INITIAL RE/RI                     =',G14.5/
     1        ' INITIAL TE/TI                     =',G14.5/
     1        ' PHI -- REACTION PARAMETER         =',G14.5/)
C
C     NOW COMPUTE THE DEPENDENT VARIABLES
C
      SHKMC2 = SHKMAC**2
C
      IF (IGAS .EQ. 1) THEN
C        USE PERFECT GAS MODEL, CONCTANT GAMMA ON BOTH SIDES
         READ (INSHOC,*) AMWTA
         READ (INSHOC,*) GAMMAI
         RGAS   = UGASFL/AMWTA
         ALPHAI = 0.
         ALPHAE = 0.
         GOTO 105
      ENDIF
```

```
C
      IF (PHI .GE. 0.) GOTO 100

C     COMPUTE THE EQUILIBRIUM VALUES ACROSS THE MOVING SHOCK

C     EQUILIBRIUM VALUES AT INLET
      CALL EQUCAL(TEMPI,THETAD,RHOI,RHOD,ETRAT,RHORAT,ALPHAI)

      PRESSI = RHOI*(1.+ALPHAI)*RGAS*TEMPI
      ALPIP4 = ALPHAI + 4.
      ALPIP1 = ALPHAI + 1.

C     COMPUTE THE DIMENSIONAL DENSITY AND TEMPERATURE AT EXIT
C     ITERATIONS MAY BE NEEDED FOR THIS CASE

20    ALPEP4 = ALPHAE + 4.
      ALPEP1 = ALPHAE + 1.

      DENOT  = 1. + SHKMC2*ALPEP1*(1.-REBRI**2)/6.
      ANUET  = ALPIP4 + THETAD/TEMPI*(ALPHAI-ALPHAE)
      TEBTI  = ANUET/(ALPEP4*DENOT)
      TEMPE = TEMPI*TEBTI
      ANUER = ALPIP1/ALPEP1/TEBTI
      DENOR = 1. + SHKMC2*ALPEP4*(1.-REBRI)/3.
      REBRI = (ANUER/DENOR)
      RHOE  = RHOI*REBRI
      ALPHAP = ALPHAE

C     COMPUTE THE EQUILIBRIUM VALUES AT EXIT

      CALL EQUCAL(TEMPE,THETAD,RHOE,RHOD,ETRAT,RHORAT,ALPHAE)

      COMPA = ABS(ALPHAE - ALPHAP)
      ALPHAE = (ALPHAE+10.*ALPHAP)/11.
      ALPHAE = (ALPHAE+IALP*ALPHAP)/(1.+IALP)

      WRITE(6,*) ' ALPHAE =',ALPHAE
      WRITE(6,*) ' COMPA  =',COMPA
      WRITE(6,*) ' TEMPE  =',TEMPE
      WRITE(6,*) ' RHOE   =',RHOE
      READ(5,*) IG
      IF (IG .EQ. 1) GOTO 20
C     IF (COMPA .GT. 1.E-5) GOTO 20

      PRESSE = RHOE*(1.+ALPHAE)*RGAS*TEMPE
      GAMMAE = ALPEP4/3.
      GAMMAI = ALPIP4/3.
      SOUNDE = GAMMAE*PRESSE/RHOE
      SOUNDE = SQRT (SOUNDE)
      US     = SHKMAC*SOUNDE
      WI     = REBRI*US
      UCOMPI = US - WI
      UE     = 0.
      SOUNDI = GAMMAI*PRESSI/RHOI
      SOUNDI = SQRT (SOUNDI)

      GOTO 200
```

```
100     GAMMAI = (4. +ALPHAI)/3.
        ALPHAE = ALPHAI
105     GM1    = GAMMAI - 1.
        GP1    = GAMMAI + 1.
        BIGGAM = GP1/GM1
        PIBPE  = (2.*GAMMAI*SHKMC2 - GM1)/GP1
        RIBRE  = (BIGGAM*PIBPE+1.)/(BIGGAM+PIBPE)

        IF (IALP .NE. 0) THEN
C          USE TEMPI AND RHOI AS INLET VALUES FOR REFERENCE CONDITIONS
           PRESSI = RHOI*(1.+ALPHAI)*RGAS*TEMPI
           PRESSE = PRESSI/PIBPE
           RHOE   = RHOI/RIBRE
           TEMPE  = PRESSE/((1.+ALPHAI)*RHOE*RGAS)
        ELSE
C          USE TEMPE AND RHOE AS EXIT VALUES FOR REFERENCE CONDITIONS
           TEMPE  = TEMPI
           RHOE   = RHOI
           RHOI   = RHOE*RIBRE
           PRESSE = RHOE*(1.+ALPHAE)*RGAS*TEMPE
           PRESSI = PRESSE*PIBPE
           TEMPI  = PRESSI/((1.+ALPHAI)*RHOI*RGAS)
        ENDIF

C       MACH NUMBER BEFORE THE SHOCK

        AMBEFO = SQRT(TEMPE/TEMPI)*2.*(SHKMC2-1.)/(SHKMAC*GP1)
        SOUNDI = GAMMAI*PRESSI/RHOI
        SOUNDI = SQRT (SOUNDI)
        UCOMPI    = AMBEFO*SOUNDI
        UE    = 0.
C
C       COMPUTE THE NORMALIZED QUANTITIES

200     CONTINUE

C       BASE REFERENCE VALUES ON EXIT
        RHORFL = RHOE
        PRESFL = PRESSE
        ALPHAR = ALPHAE

        ONEPA1 = 1. + ALPHAR
C       ONEPA1 IS 1 + ALPHAR
C       COMPUTE THE REFERENCE TEMPERATURE
        TREFFL = PRESFL/(ONEPA1*RHORFL*RGAS)
C       COMPUTE THE REFERENCE VELOCITY
        UREFFL = SQRT(PRESFL/RHORFL)
        RHOINL = RHOI
        TINLET = TEMPI
        PINLET = PRESSI
        RHOEXT = RHOE
        PEXIT  = PRESSE
        TEXIT  = TEMPE
        SOUNDI = GAMMAI*PRESSI/RHOI
        SOUNDI = SQRT (SOUNDI)
C
```

```
C       NORMALIZE ALL VALUES
C
        RHOI   = RHOI/RHORFL
        PRESSI = PRESSI/PRESFL
        TEMPI  = TEMPI/TREFFL
        RHOE   = RHOE/RHORFL
        PRESSE = PRESSE/PRESFL
        TEMPE  = TEMPE/TREFFL

        THETAD = THETAD/TREFFL
        RHOD   = RHOD/RHORFL
        UBEFOR = UCOMPI/UREFFL
        AMBEFO = UCOMPI/SOUNDI
C
C       WRITE DOWN THE NORMALIZED VALUES IN MOSHOC.OUT FOR RECORD.
C
        WRITE(IOSHOC,210) RHOE,PRESSE,TEMPE, RHOI,PRESSI,TEMPI,
     1                    RHORFL,PRESFL,TREFFL, RHOINL,PINLET,TINLET,
     1                    RHOEXT,PEXIT,TEXIT,
     2                    GAMMAI,ALPHAI,UBEFOR,AMBEFO,RHOD,THETAD
210        FORMAT (' RHOE  =',G14.5,5X,'PRESSE=',G14.5,5X,'TEMPE =',G14.5/
     1             ' RHOI  =',G14.5,5X,'PRESSI=',G14.5,5X,'TEMPI =',G14.5/
     2             ' RHORFL=',G14.5,5X,'PRESFL=',G14.5,5X,'TREFFL=',G14.5/
     3             ' RHOINL=',G14.5,5X,'PINLET=',G14.5,5X,'TEMPI=',G14.5/
     3             ' RHOEXT=',G14.5,5X,'PEXIT =',G14.5,5X,'TEXIT=',G14.5/
     4             ' GAMMAI=',G14.5,5X,'ALPHAI=',G14.5,5X,'UBEFOR=',G14.5/
     5             ' MBEFOR=',G14.5,5X,'RHOD  =',G14.5,5X,'THETAD=',G14.5/)
C
C       WRITE DOWN THE VALUES IN MOSHOC.IOT SO THAT THESE CAN BE READ LATER
C       BY A GRID GENERATOR PROGRAM LIKE BEPIPE.FOR
C
        AGAS = IGAS
        WRITE(IOPIPE,220) ALPHAI
        WRITE(IOPIPE,220) ALPHAE
        WRITE(IOPIPE,220) ALPHAR
        WRITE(IOPIPE,220) RHOI
        WRITE(IOPIPE,220) RHOE
        WRITE(IOPIPE,220) UBEFOR
        WRITE(IOPIPE,220) UE
        WRITE(IOPIPE,220) PRESSI
        WRITE(IOPIPE,220) PRESSE
        WRITE(IOPIPE,220) TEMPI
        WRITE(IOPIPE,220) TEMPE
        WRITE(IOPIPE,220) TREFFL
        WRITE(IOPIPE,220) RHORFL
        WRITE(IOPIPE,220) SHKMAC
        WRITE(IOPIPE,220) AGAS
        WRITE(IOPIPE,220) RGAS
        WRITE(IOPIPE,220) GAMMAI
220     FORMAT (E15.8)

C       EXAMPLE FILE : MOSHOC.DAT
C 0.20                ALPHAI
C 0.00                ALPHAE
C 4.8                 RHOI
C 4600.0              TINLET
C 6.                  SHKMAC
```

```
C 0.19              REBRI
C 0.125             TEBTI
C 0.01              PHI

      END
```

Non-equilibrium moving shock

The program LHSHOC calculates the initial conditions across a moving Lighthill non-equilibrium shock. The conditions in the quiescent gas are assumed known and the conditions behind the shock are obtained by solving an O.D.E. for the degree of dissociation (see Section 7.3.2).

```
      PROGRAM LHSHOC

      PARAMETER (MPOINT = 1000)
      DIMENSION A$(MPOINT), P$(MPOINT), R$(MPOINT), T$(MPOINT),
     1          U$(MPOINT), X$(MPOINT), IOPT$(1)  , N$(1)     ,
     2          E1TAX(6)
      CHARACTER MTITLE*80 , PLTITL*80 , E1TAX*12  , YESNO*1
      DATA E1TAX/ 'ALPHA      ', 'PRESSURE   ', 'DENSITY    ',
     1            'TEMPERATURE', 'VELOCITY   ', '           '/

C*********************************************************************
C
C     THIS PROGRAM GENERATES THE NON-EQUILIBRIUM CONDITIONS FOR A
C     MOVING SHOCK.  CONDITIONS AT THE INLET (BEHIND SHOCK) ARE
C     ASSUMED TO BE KNOWN AND READ FROM FILE LHSHOC.DAT.  THE VELOCITIES
C     WI AND WE DENOTE THE VALUES FOR STANDING NORMAL SHOCK, WHEREAS UI
C     AND UE DENOTE THE CORRESPONDING VALUES FOR A MOVING SHOCK.
C
C     SUBROUTINES CALLED: EQUCAL     UTILITY ROUTINE
C                         GR_INIT    GRAFIC  ROUTINE
C                         GR_LINE    GRAFIC  ROUTINE
C
C*********************************************************************
C
C     INITIALIZATION
C
      INSHOC  = 51
      JPRINT  = 6
      IOSHOC  = 8
      MOSHOC  = 9
C
C     VALUES FOR OXYGEN ARE ASSUMED HERE FOR THE LIGHTHILL MODEL
C
      THETAD  = 59500.
      RHOD    = 150.E03
      UGASFL  = 8314.3
```

```
      AMWTA    = 16.0
      RGAS-    = 0.5*UGASFL/AMWTA
      HTFORM   = RGAS*THETAD
      ETA      = 0.
      COMPM    = 1.E-6
      NLINE    = 1
      IOPT$(1) = 2
      MTITLE   = 'NON-EQUILIBRIUM  MOVING SHOCK'
      INDGR    = 21
      AGAS     = 2.
C
C     OPEN THE APPROPRAITE UNITS
C
      OPEN (UNIT = INSHOC, FILE = 'LHSHOC.DAT', STATUS = 'OLD')
      OPEN (UNIT = IOSHOC, FILE = 'LHSHOC.OUT', STATUS = 'NEW')
      OPEN (UNIT = MOSHOC, FILE = 'MOSHOC.IOT', STATUS = 'NEW')
C
C     INPUT THE FOLLOWING QUANTITIES FROM FILE LHSHOC.DAT
C
C        ALPHAI    INITIAL ALPHA AT INLET
C        ALPHAE    INITIAL ALPHA AT EXIT
C        RHOI      DENSITY AT INLET IN kg/m**3
C        TEMPI     TEMPERATURE AT INLET IN Kelvins
C        SHKMAC    MACH NUMBER OF THE SHOCK (BASED UPON STATION E)
C        REBRI     DENSITY RATIO (GUESS)
C        TEBTI     TEMPERATURE RATIO (GUESS)
C        PHI       REACTION PARAMETER (<0 MEANS EQUILIBRIUM)
C        IALP      WEIGHT FACTOR FOR EQUILIBRIUM ALPHA, NORMAL VALUE=0
C                   AVERAGE VALUE=1, FOR VERY SLOW CONVERGENCE VALUE=10
C        XSHOC     DISTANCE MEASURE FOR SHOCK LOCATION
C        XMAX      MAXIMUM DISTANCE
C        XMIN      MINIMUM DISTANCE
C        NPOS      NUMBER OF POINTS ON THE EXIT SIDE (NONE IS NEEDED !)
C        NNEG      NUMBER OF POINTS ON THE INLET SIDE

      READ (INSHOC,*) ALPHAI
      READ (INSHOC,*) ALPHAE
      READ (INSHOC,*) RHOI
      READ (INSHOC,*) TEMPI
      READ (INSHOC,*) SHKMAC
      READ (INSHOC,*) REBRI
      READ (INSHOC,*) TEBTI
      READ (INSHOC,*) PHI
      READ (INSHOC,*) IALP
      READ (INSHOC,*) XSHOC
      READ (INSHOC,*) XMAX
      READ (INSHOC,*) XMIN
      READ (INSHOC,*) NPOS
      READ (INSHOC,*) NNEG

C     COMPUTE THE STEP SIZES ON EITHER SIDES OF THE SHOCK LOCATION

      XDPOS = (XMAX - XSHOC)/(NPOS - 1)
      XDNEG = (XMIN - XSHOC)/(NNEG - 1)

C     NOW COMPUTE THE SQUARE OF MACH NUMBER
C
```

```
      SHKMC2 = SHKMAC**2
C          -
C     COMPUTE THE EQUILIBRIUM VALUES AT INLET
      CALL EQUCAL(TEMPI,THETAD,RHOI,RHOD,ETRAT,RHORAT,ALPHAI)

      PRESSI = RHOI*(1.+ALPHAI)*RGAS*TEMPI
      ALPIP4 = ALPHAI + 4.
      ALPIP1 = ALPHAI + 1.

C     COMPUTE THE DIMENSIONAL DENSITY AND TEMPERATURE AT EXIT
C     ITERATIONS MAY BE NEEDED FOR THIS CASE

10    ALPEP4 = ALPHAE + 4.
      ALPEP1 = ALPHAE + 1.

      DENOT  = 1. + SHKMC2*ALPEP1*(1.-REBRI**2)/6.
      ANUET  = ALPIP4 + THETAD/TEMPI*(ALPHAI-ALPHAE)
      TEBTI  = ANUET/(ALPEP4*DENOT)
      TEMPE  = TEMPI*TEBTI
      ANUER  = ALPIP1/ALPEP1/TEBTI
      DENOR  = 1. + SHKMC2*ALPEP4*(1.-REBRI)/3.
      REBRI  = (ANUER/DENOR)
      RHOE   = RHOI*REBRI
      ALPHAP = ALPHAE

C     COMPUTE THE EQUILIBRIUM VALUES AT EXIT

      CALL EQUCAL(TEMPE,THETAD,RHOE,RHOD,ETRAT,RHORAT,ALPHAE)

      COMPA = ABS(ALPHAE - ALPHAP)
      ALPHAE = (ALPHAE+10.*ALPHAP)/11.
      ALPHAE = (ALPHAE+IALP*ALPHAP)/(1.+IALP)

      IF (COMPA .GT. COMPM) GOTO 10
      WRITE(6,*) ' ALPHAE =',ALPHAE
      WRITE(6,*) ' COMPA  =',COMPA
      WRITE(6,*) ' TEMPE  =',TEMPE
      WRITE(6,*) ' RHOE   =',RHOE
      READ(5,*) IG
      IF (IG .EQ. 1) GOTO 10

      PRESSE = RHOE*(1.+ALPHAE)*RGAS*TEMPE
      GAMMAE = ALPEP4/3.
      GAMMAI = ALPIP4/3.
      SOUNDE = GAMMAE*PRESSE/RHOE
      SOUNDE = SQRT (SOUNDE)
      US     = SHKMAC*SOUNDE
      WI     = REBRI*US
      WE     = US
      UCOMPI = US - WI
      UE     = 0.
      SOUNDI = GAMMAI*PRESSI/RHOI
      SOUNDI = SQRT (SOUNDI)

C     BASE THE REFERENCE VALUES ON EXIT STATION
C     ONEPA1 IS 1 + ALPHAR
```

```
      RHORFL = RHOE
      PRESFL = PRESSE
      ALPHAR = ALPHAE
      ONEPA1 = 1. + ALPHAR
C     COMPUTE THE REFERENCE TEMPERATURE
      TREFFL = PRESFL/(ONEPA1*RHORFL*RGAS)
C     COMPUTE THE REFERENCE VELOCITY
      UREFFL = SQRT(PRESFL/RHORFL)

C     SAVE THE DIMENSIONAL VALUES

      RHOINL = RHOI
      TINLET = TEMPI
      PINLET = PRESSI
      RHOEXT = RHOE
      PEXIT  = PRESSE
      TEXIT  = TEMPE
      SOUNDI = GAMMAI*PRESSI/RHOI
      SOUNDI = SQRT (SOUNDI)
C
C     COMPUTE THE NORMALIZED QUANTITIES
C
      RHOI   = RHOI/RHORFL
      PRESSI = PRESSI/PRESFL
      TEMPI  = TEMPI/TREFFL
      RHOE   = RHOE/RHORFL
      PRESSE = PRESSE/PRESFL
      TEMPE  = TEMPE/TREFFL
      WI     = WI/UREFFL
      WE     = WE/UREFFL
      THETAD = THETAD/TREFFL
      RHOD   = RHOD/RHORFL
      UBEFOR = UCOMPI/UREFFL
C     MACH NUMBER BEFORE THE SHOCK
      AMBEFO = UCOMPI/SOUNDI
      HTFORM = HTFORM/(UREFFL*UREFFL)

C     FOR EQUILIBRIUM FLOW NOTHING ELSE IS NEEDED
      IF (PHI .LT. 0.) GOTO 50
C
C     COMPUTE CONSTANTS OF SHOCK MOTION FOR MASS, MOMENTUM AND ENERGY
C
      CONCON = RHOE*WE
      CONMOM = CONCON*WE + PRESSE
      CONENG = (4.+ALPHAE)/(1.+ALPHAE)*PRESSE/RHOE + ALPHAE*HTFORM +
     1         0.5*WE*WE

C     COMPUTE THE CONDITIONS AT STATION S (JUST AFTER THE FRONTAL SHOCK)
      ALPHAS = ALPHAE
      GAMMAS = (4. +ALPHAS)/3.
      GM1    = GAMMAS - 1.
      GP1    = GAMMAS + 1.
      BIGGAM = GP1/GM1
      PSBPE  = (2.*GAMMAS*SHKMC2 - GM1)/GP1
      RSBRE  = (BIGGAM*PSBPE+1.)/(BIGGAM+PSBPE)
      TSBTE  = PSBPE/RSBRE
      PRESSS = PSBPE*PRESSE
```

```
      RHOS   = RSBRE*RHOE
      TEMPS  = TSBTE*TEMPE
      WS     = CONCON/RHOS
C
C     COLLECT CONSTANT DATA FOR THE QUIESCENT SIDE OF THE SHOCK
C
      NPOINT = 0
      DO 20 IPOS = NPOS, 1, -1
        NPOINT = NPOINT + 1
        X$(NPOINT) = XSHOC + (IPOS-1)*XDPOS
        A$(NPOINT) = ALPHAE
        P$(NPOINT) = PRESSE
        R$(NPOINT) = RHOE
        T$(NPOINT) = TEMPE
        U$(NPOINT) = WE - WE
20    CONTINUE

C     X$(NPOINT) = XSHOC
C     A$(NPOINT) = ALPHAS
C     P$(NPOINT) = PRESSS
C     R$(NPOINT) = RHOS
C     T$(NPOINT) = TEMPS
C     U$(NPOINT) = WE - WS
C
C     NUMERICALLY INTEGRATE THE ODE FOR MASS FRACTION OF ATOMS AND
C     COLLECT DATA IN THE NON-LINEAR SIDE BEHIND THE SHOCK
C
      ALPHA = ALPHAS
      RHO   = RHOS
      PRESS = PRESSS
      TEMP  = TEMPS
      WVELO = WS

      DO 30 INEG = 1, NNEG-1
        NPOINT = NPOINT + 1
        X$(NPOINT) = XSHOC + INEG*XDNEG
        TETA = TEMP**ETA
        TRAT = -THETAD/TEMP
        RRAT = RHO/RHOD
        SBRAC = (1.-ALPHA)*EXP(TRAT) - RRAT*ALPHA*ALPHA
        FACTOR = PHI*TETA*RHO/WVELO
        SOURCE = FACTOR*SBRAC
        ALPHAN = ALPHA - SOURCE*XDNEG
        BIGA = 7. + ALPHAN
        BIGB = 2.*(4.+ALPHAN)*CONMOM/CONCON
        BIGC = 2.*(1.+ALPHAN)*(CONENG-ALPHAN*HTFORM)
        DISCRI = SQRT(BIGB**2-4.*BIGA*BIGC)
        DENO = 2.*BIGA
        ROOT1 = (BIGB-DISCRI)/DENO
C       ROOT2 = (BIGB+DISCRI)/DENO
        WVELO = ROOT1
        RHO = CONCON/WVELO
        PRESS = CONMOM - CONCON*WVELO
        PRAT = PRESS/PRESSS
        RRAT = RHO/RHOS
        ARAT = (1.+ALPHAN)/(1.+ALPHAS)
        TEMP  = PRAT*TEMPS/(ARAT*RRAT)
```

```
          ALPHA = ALPHAN
          A$(NPOINT) = ALPHA
          P$(NPOINT) = PRESS
          R$(NPOINT) = RHO
          T$(NPOINT) = TEMP
          U$(NPOINT) = WE - WVELO
30      CONTINUE
C
C       COMPUTE DISSOCIATION THICKNESS BASED UPON 99 % VALUE
C
        ALPCOM = 0.99*ALPHAI
        XDISSO = -999.
        DO 40 IP = 1, NPOINT-1
          IF (ALPCOM .GE. A$(IP) .AND. ALPCOM .LE. A$(IP+1)) THEN
            ARAT   = (ALPCOM - A$(IP))/(A$(IP+1) - A$(IP))
            DELX   = X$(IP+1) - X$(IP)
            XDISSO = X$(IP) + DELX*ARAT
            XDISSO = ABS(XDISSO - XSHOC)
            GOTO 50
          ENDIF
40      CONTINUE
C
C       WRITE DOWN THESE VALUES IN LHSHOC.OUT FOR THE PURPOSE OF KEEPING
C       A RECORD
C
50      WRITE(IOSHOC,60) ALPHAI, ALPHAE, RHOINL, TINLET, SHKMAC,
     1                   REBRI , TEBTI , PHI    , XDISSO
60      FORMAT (' INITIAL ALPHA AT INLET             =',G14.5/
     1          ' INITIAL ALPHA AT EXIT              =',G14.5/
     2          ' RHO INLET                          =',G14.5/
     3          ' TEMPERAURE INLET                   =',G14.5/
     4          ' SHOCK MACH NUMBER                  =',G14.5/
     5          ' INITIAL RE/RI                      =',G14.5/
     6          ' INITIAL TE/TI                      =',G14.5/
     7          ' PHI -- REACTION PARAMETER          =',G14.5/
     7          ' DISSOCIATION THICKNESS             =',G14.5/)
C
C
        WRITE(IOSHOC,70) RHOE,PRESSE,TEMPE, RHOI,PRESSI,TEMPI,
     1                   RHORFL,PRESFL,TREFFL, RHOINL,PINLET,TINLET,
     2                   RHOEXT,PEXIT,TEXIT,
     3                   GAMMAI,ALPHAI,UBEFOR,AMBEFO,RHOD,THETAD
70        FORMAT (' RHOE  =',G14.5,5X,'PRESSE=',G14.5,5X,'TEMPE =',G14.5/
     1            ' RHOI  =',G14.5,5X,'PRESSI=',G14.5,5X,'TEMPI =',G14.5/
     2            ' RHORFL=',G14.5,5X,'PRESFL=',G14.5,5X,'TREFFL=',G14.5/
     3            ' RHOINL=',G14.5,5X,'PINLET=',G14.5,5X,'TEMPI=',G14.5/
     3            ' RHOEXT=',G14.5,5X,'PEXIT =',G14.5,5X,'TEXIT=',G14.5/
     4            ' GAMMAI=',G14.5,5X,'ALPHAI=',G14.5,5X,'UBEFOR=',G14.5/
     5            ' MBEFOR=',G14.5,5X,'RHOD  =',G14.5,5X,'THETAD=',G14.5/)
C
C       WRITE DOWN THE VALUES IN MOSHOC.IOT SO THAT THESE CAN BE READ LATER
C       BY A GRID GENERATOR PROGRAM LIKE BEPIPE.FOR
C
        WRITE(MOSHOC,80) ALPHAI
        WRITE(MOSHOC,80) ALPHAE
        WRITE(MOSHOC,80) ALPHAR
        WRITE(MOSHOC,80) RHOI
```

```
      WRITE(MOSHOC,80) RHOE
      WRITE(MOSHOC,80) UBEFOR
      WRITE(MOSHOC,80) UE
      WRITE(MOSHOC,80) PRESSI
      WRITE(MOSHOC,80) PRESSE
      WRITE(MOSHOC,80) TEMPI
      WRITE(MOSHOC,80) TEMPE
      WRITE(MOSHOC,80) TREFFL
      WRITE(MOSHOC,80) RHORFL
      WRITE(MOSHOC,80) SHKMAC
      WRITE(MOSHOC,80) AGAS
80    FORMAT (E15.8)
C
C     WRITE DOWN THE VALUES AT ALL X-LOCATIONS IN MOSHOC.IOT SO THAT
C     THESE CAN BE READ LATER BY A PRE-EMBEDDING ROUTINE OF THE STAR
C     CODE
C
      WRITE(MOSHOC,*) NPOINT
      DO 90 IP = 1, NPOINT
        WRITE(MOSHOC,100) X$(IP), A$(IP), P$(IP), R$(IP), T$(IP), U$(IP)
90    CONTINUE
100   FORMAT(6E15.7)

C     OPTION TO PLOT THE DATA

      WRITE(6,*) ' WANT TO PLOT DATA'
      READ (5,110) YESNO
110   FORMAT(A1)

      IF (YESNO .NE. 'Y' .AND. YESNO .NE. 'y') STOP ' THE END'

      CALL GR_INIT(5,6,MTITLE)
      N$(1) = NPOINT
      PLTITL(1:9) = 'DISTANCE'
120   WRITE(6,130)
130   FORMAT(1X,'THE FOLLOWING VARIABLES CAN BE PLOTTED VERSUS X'/
     1       5X,'1. DEGREE OF DISSOCIATION'/
     2       5X,'2. PRESSURE              '/
     3       5X,'3. DENSITY               '/
     4       5X,'4. TEMPERATURE           '/
     5       5X,'5. VELOCITY              '/
     6       5X,'6. EXIT                  '/ ' ===> ',$)

      READ (5,*) IPLOT
      PLTITL(10:22) = E1TAX(IPLOT)

      IF (IPLOT .EQ. 1) THEN
        CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,A$,N$)
      ELSE IF (IPLOT .EQ. 2) THEN
        CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,P$,N$)
      ELSE IF (IPLOT .EQ. 3) THEN
        CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,R$,N$)
      ELSE IF (IPLOT .EQ. 4) THEN
        CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,T$,N$)
      ELSE IF (IPLOT .EQ. 5) THEN
        CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,U$,N$)
      ENDIF
```

```
      WRITE (6,*) ' WANT TO PLOT MORE'
      READ (5,110) YESNO

      IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y') GOTO 120

C     EXAMPLE FILE : LHSHOC.DAT
C 0.40              ALPHAI
C 0.2               ALPHAE
C 3.82              RHOI
C 5000.0            TINLET
C 2.35              SHKMAC
C 0.263             REBRI
C 0.825             TEBTI
C 1.E4              PHI
C 0                 IALP
C-0.5               XSHOC
C 1.                XMAX
C-1.                XMIN
C10                 NPOS
C100                NNEG

      STOP ' THE END'
      END
```

D.1.2 Initial grid generators

The grid generators for the circular arc bump case (Section 8.2.1 to 8.2.3) and a bend duct (Section 8.2.4 to 8.2.5) are presented here.

Circular arc cascade

The program NISHOC generates the initial grid for a circular arc cascade on a lower channel wall. The initial conditions are read from the output of a previous initial condition generator program.

```
      PROGRAM NISHOC

      PARAMETER (MXX = 500 ,   MYY = 500, MNODG2 = 16000)

      DIMENSION XSOUTH(MXX), XEAST(MYY), XNORTH(MXX), XWEST(MYY),
     1          YSOUTH(MXX), YEAST(MYY), YNORTH(MXX), YWEST(MYY),
     2          DISTW(MXX),  DISTE(MXX), GEOMG2(2,MNODG2),
     3          IBNDG2(5,500)  , DPENG2(5)
```

```
      DIMENSION A$(MXX), P$(MXX), R$(MXX), T$(MXX), U$(MXX),
     1          X$(MXX), IOPT$(1), N$(1) , E1TAX(6)
      CHARACTER MTITLE*80 , PLTITL*80 , E1TAX*12  , YESNO*1
      DATA E1TAX/ 'ALPHA      ', 'PRESSURE   ', 'DENSITY    ',
     1            'TEMPERATURE', 'VELOCITY   ', '           '/

C*********************************************************************
C
C       THIS PROGRAM GENERATES THE BOUNDARIES OF THE COMPUTATION DOMAIN
C       FOR THE NI BUMP PROBLEM INVOLVING A MOVING SHOCK.  IT ALSO READS
C       THE INITIAL CONDITIONS FOR THIS CASE FROM FILE MOSHOC.IOT FOR A
C       PERFECT OR A LIGHT-HILL GAS.  THIS FILE MAY HAVE BEEN GENERATED
C       BY EITHER MOSHOC.FOR OR LHSHOC.FOR.  THE VARIABLES FOR THE CURRENT
C       GEOMETRY ARE READ FROM THE FILE NISHOC.DAT THAT CONTAINS THE
C       LOCATION OF INLET, EXIT, BOTTOM, AND TOP WALLS.  THE BUMP IS
C       ASSUMED BETWEEN X=0 AND X=1 ON LOWER WALL.  OTHER PARAMETERS IN
C       NISHOC.DAT INCLUDE THE SIZE (PERCENTAGE POINT) OF THE BUMP AND THE
C       POSITION OF THE SHOCK AT TIME T=0.  OTHER INFORMATION INCLUDES THE
C       NUMBER OF DATA POINTS ALONG X AND Y DIRECTIONS.
C       THE GRID IS GENERATED BY AN ALGEBRAIC CONSTRUCTION.  FINALLY THE
C       PROGRAM INITIALIZES THE DEPENDENT VARIABLES OVER ALL THE NODES.
C       THE OUTPUT FILE INPUTG.DAT CONTAINS GRID INFORMATION SUCH AS X
C       AND Y COORDINATES AT EACH NODE WHEREAS THE OUTPUT FILE INPUTD.DAT
C       CONTAINS DEPENDENT VARIABLE INFORMATION SUCH AT THESE NODES.
C
C       SUBROUTINES CALLED: ERRORM      UTILITY ROUTINE
C                           GR_INIT     GRAFIC  ROUTINE
C                           GR_LINE     GRAFIC  ROUTINE
C
C*********************************************************************
C
C       INITIALIZATION
C
        INTUBE = 51
        INPUTG = 52
        INPUTD = 54
        JPRINT = 6
        IOSHOC = 8
        MOSHOC = 9
        MTITLE = 'NON-EQUILIBRIUM  MOVING SHOCK'
C
C       OPEN THE APPROPRAITE UNITS
C
        OPEN (UNIT = MOSHOC, FILE = 'MOSHOC.IOT', STATUS = 'OLD')
        OPEN (UNIT = INTUBE, FILE = 'NISHOC.DAT', STATUS = 'OLD')
        OPEN (UNIT = INPUTG, FILE = 'INPUTG.DAT', STATUS = 'NEW')
        OPEN (UNIT = INPUTD, FILE = 'INPUTD.DAT', STATUS = 'NEW')
        OPEN (UNIT = IOSHOC, FILE = 'NISHOC.OUT', STATUS = 'NEW')
C
C       INPUT THE FOLLOWING QUANTITIES FROM FILE NISHOC.DAT
C
C           XMIN      X-DISTANCE MEASURE AT THE INLET
C           XMAX      X-DISTANCE MEASURE AT THE EXIT
C           YMAX      Y-DISTANCE MEASURE AT THE TOP WALL
C           XCONTA    X-DISTANCE FOR THE CONTACT SURFACE
C           NXRECT    NUMBER OF NODES ALONG X-DIRECTION
```

```
C          NYRECT    NUMBER OF NODES ALONG Y-DIRECTION
C          PERCEN    PERCENTAGE POINT OF THE BUMP
C
      READ (INTUBE,*) XMIN
      READ (INTUBE,*) XMAX
      READ (INTUBE,*) YMAX
      READ (INTUBE,*) XCONTA
      READ (INTUBE,*) NXRECT
      READ (INTUBE,*) NYRECT
      READ (INTUBE,*) PERCEN
C
C     READ THE FOLLOWING VALUES FROM MOSHOC.IOT
C
C          ALPHAI    INITIAL ALPHA AT INLET
C          ALPHAE    INITIAL ALPHA AT EXIT
C          ALPHAR    REFERENCE ALPHA
C          RHOI      INLET NON-DIMENSIONAL DENSITY
C          RHOE      EXIT NON-DIMENSIONAL DENSITY
C          UCOMPI    INLET NON-DIMENSIONAL VELOCITY
C          UCOMPE    EXIT NON-DIMENSIONAL VELOCITY
C          PRESSI    INLET NON-DIMENSIONAL PRESSURE
C          PRESSE    EXIT NON-DIMENSIONAL PRESSURE
C          TEMPI     INLET NON-DIMENSIONAL TEMPERATURE
C          TEMPE     EXIT NON-DIMENSIONAL TEMPERATURE
C          TREFFL    REFERENCE TEMPERATURE
C          SHKMAC    MACH NUMBER OF THE SHOCK
C          AGAS IGAS PARAMETER INDICATING TYPE OF GAS
C                    1: PERFECT     0: LIGHTHILL
C                    2: LIGHTHILL GAS FOR NON-EQUILIBRIUM

      READ (MOSHOC,10) ALPHAI
      READ (MOSHOC,10) ALPHAE
      READ (MOSHOC,10) ALPHAR
      READ (MOSHOC,10) RHOI
      READ (MOSHOC,10) RHOE
      READ (MOSHOC,10) UBEFOR
      READ (MOSHOC,10) UE
      READ (MOSHOC,10) PRESSI
      READ (MOSHOC,10) PRESSE
      READ (MOSHOC,10) TEMPI
      READ (MOSHOC,10) TEMPE
      READ (MOSHOC,10) TREFFL
      READ (MOSHOC,10) RHORFL
      READ (MOSHOC,10) SHKMAC
      READ (MOSHOC,10) AGAS
10    FORMAT (E15.8)
C
      IGAS = NINT(AGAS)
      NTOTAL = 5
      TFACTR = 3.

      IF (IGAS .EQ. 1) THEN
C        USE PERFECT GAS MODEL FOR THE SHOCK
C        READ SOME MORE VALUES FROM MOSHOC.IOT
         READ (MOSHOC,10) RGAS
         READ (MOSHOC,10) GAMMAI
         NTOTAL = 4
```

```
            ALPHAI = 0.
            ALPHAE = 0.
            ALPHAR = 0.
            TFACTR = 1./(GAMMAI-1.)
          ENDIF
C
          IF (IGAS .EQ. 2) THEN
C           USE LIGHTHILL GAS MODEL FOR THE NON-EQUILIBRIUM SHOCK
C           READ DEPENDENT VARIABLE VALUES FROM MOSHOC.IOT
            READ(MOSHOC,*) NPOINT
            DO 20 IP = 1, NPOINT
              READ(MOSHOC,30) X$(IP),A$(IP),P$(IP),R$(IP),T$(IP),U$(IP)
20          CONTINUE
          ENDIF
30        FORMAT(6E15.7)
C
C         WRITE DOWN THESE VALUES IN NISHOC.OUT FOR THE PURPOSE OF KEEPING
C         A RECORD
C
          WRITE(IOSHOC,40) ALPHAI, ALPHAE, ALPHAR, RHOI, RHOE, TEMPI,
     1         TEMPE, PRESSI, PRESSE, TREFFL, RHORFL, XMIN, XMAX, YMAX,
     2         XCONTA, PERCEN, NXRECT, NYRECT, SHKMAC
40        FORMAT (' INITIAL ALPHA AT INLET             =',G14.5/
     1          ' INITIAL ALPHA AT EXIT              =',G14.5/
     1          ' REFERENCE ALPHA                    =',G14.5/
     1          ' RHOI                               =',G14.5/
     1          ' RHOE                               =',G14.5/
     1          ' TEMPI                              =',G14.5/
     1          ' TEMPE                              =',G14.5/
     1          ' PRESSI                             =',G14.5/
     1          ' PRESSE                             =',G14.5/
     1          ' TREFFL                             =',G14.5/
     1          ' RHORFL                             =',G14.5/
     1          ' X-DISTANCE MEASURE AT THE INLET    =',G14.5/
     1          ' X-DISTANCE MEASURE AT THE EXIT     =',G14.5/
     1          ' Y-DISTANCE MEASURE AT THE TOP WALL =',G14.5/
     1          ' X-DISTANCE FOR THE CONTACT SURFACE =',G14.5/
     1          ' PERCENTAGE POINT OF THE BUMP       =',G14.5/
     1          ' NUMBER OF NODES ALONG X-DIRECTION  =',I5/
     1          ' NUMBER OF NODES ALONG Y-DIRECTION  =',I5/
     1          ' SHOCK MACK NUMBER                  =',G14.5/)

C         CHECK FOR OVERFLOW IN BOUNDARY NODE ARRAYS

          IF (NXREXT .GT. MXX) THEN
            ERR1 = NXRECT
            ERR2 = MXX
            CALL ERRORM (4,'G2RECT','NXRECT',ERR1,'MXX   ',ERR2,JPRINT,
     1        'NUMBER OF NODES EXCEEDS ITS LIMIT')
          ENDIF

          IF (NYREXT .GT. MYY) THEN
            ERR1 = NYRECT
            ERR2 = MYY
            CALL ERRORM (4,'G2RECT','NYRECT',ERR1,'MYY   ',ERR2,JPRINT,
     1        'NUMBER OF NODES EXCEEDS ITS LIMIT')
          ENDIF
```

```
C
C       COMPUTE THE STEP SIZES ON EACH SIDE
C
        YMIN   = 0.
        DELX   = (XMAX-XMIN)/(NXRECT-1)
        DELY   = (YMAX-YMIN)/(NYRECT-1)
C
C       COMPUTE (X,Y) COORDINATES FOR INFLOW/OUTFLOW BOUNDARIES
C
        DO 50 IY = 1, NYRECT
           XWEST(IY) = XMIN
           XEAST(IY) = XMAX
           YWEST(IY) = YMIN + (IY-1.)*DELY
           YEAST(IY) = YWEST(IY)
50      CONTINUE
C
C       COMPUTE (X,Y) COORDINATES FOR TOP/BOTTOM BOUNDARIES
C
        DO 60 IX = 1, NXRECT
           XSOUTH(IX) = XMIN + (IX-1)*DELX
           XNORTH(IX) = XSOUTH(IX)
           YSOUTH(IX) = YMIN
           YNORTH(IX) = YMAX
60      CONTINUE
C
C       COMPUTE THE RADIUS OF THE CIRCULAR ARC BUMP
C
        IF (PERCEN .NE. 0.) THEN
           RADIUS = (4.*PERCEN*PERCEN + 1.)/(8.*PERCEN)
        ELSE
           RADIUS = 0.
        ENDIF
        R2     = RADIUS*RADIUS
C
C       CORRECT THE Y-COORDINATES FOR THE LOWER CHANNEL WALL WHERE THE
C       BUMP IS PLACED
C
        DO 70 IX = 1, NXRECT
           XX = XSOUTH(IX)
C
           IF (XX .GT. 0. .AND. XX .LT. 1.) THEN
              XX         = XX - 0.5
              YSOUTH(IX) = PERCEN - RADIUS + SQRT(R2 - XX*XX)
           ENDIF
70      CONTINUE
C
C       CORRECT THE Y-COORDINATES FOR THE OUTFLOW WALL IF A CURTAILED
C       DOMAIN IS USED
C
        IF (XSOUTH(NXRECT) .LT. 1.) THEN
           YMINH = YSOUTH(NXRECT)
           DELYH = (YMAX-YMINH)/(NYRECT-1)
           DO 75 IY = 1, NYRECT
              YEAST(IY) = YMINH + (IY-1.)*DELYH
75         CONTINUE
        ENDIF
C
```

```
C       READ THE BOUNDARY CONDITION INDICATORS FROM NISHOC.DAT
C       DIRECHLET : 2      REFLECTION : 1     WALL : 3
C
        READ (INTUBE,*) IBCSW
        READ (INTUBE,*) IBCS
        READ (INTUBE,*) IBCSE
        READ (INTUBE,*) IBCE
        READ (INTUBE,*) IBCNE
        READ (INTUBE,*) IBCN
        READ (INTUBE,*) IBCNW
        READ (INTUBE,*) IBCW
C
C
C           ------------
C           NOMENCLATURE
C           ------------
C
C       NUMBER THE COMPUTATIONAL NODES AS SHOWN IN THE FOLLOWING
C       DIAGRAM
C
C                                                          NX=NXRECT
C                                                          NY=NYRECT
C                         L  L  L                          L =NBEFNO
C                         +  +  + . . .
C                         1  2  3
C              1+(NY-1)*NX +--+--+--+--+--+--+--+ NY*NX
C                         +        NORTH          + (NY-1)*NX = L
C                         + W                  E +
C  INDJW                  + E                  A +                  INDJE
C                         + S                  S + ...
C                    ...  + S                  S +
C                  1+2*NX + T                  T + 3*NX
C                    1+NX +        SOUTH         + 2*NX
C                       1 +--+--+--+--+--+--+--+ NX
C                           2  3  ...      NX-1
C
C
C       COMPUTE THE NODE BEFORE THE FIRST NORTH ONE (L IN FIG.)
C       AND THE MAXIMUM NUMBER OF NODES

        NBEFNO = NXRECT*(NYRECT-1)
        NNODG2 = NXRECT* NYRECT
C
C       SET SOUTH AND NORTH NODE INFORMATION AS SHOWN ABOVE
C
        DO 80 IX = 1, NXRECT
           NOS           = IX
           NON           = IX + NBEFNO
           GEOMG2(1,NOS) = XSOUTH(IX)
           GEOMG2(2,NOS) = YSOUTH(IX)
           GEOMG2(1,NON) = XNORTH(IX)
           GEOMG2(2,NON) = YNORTH(IX)
80      CONTINUE
C
C       SET WEST AND EAST NODE INFORMATION AS SHOWN ABOVE
C
        DO 90 IY = 1, NYRECT
           NOW           = 1 + (IY-1)*NXRECT
```

```
            NOE           = IY*NXRECT
            GEOMG2(1,NOW) = XWEST (IY)
            GEOMG2(2,NOW) = YWEST (IY)
            GEOMG2(1,NOE) = XEAST (IY)
            GEOMG2(2,NOE) = YEAST (IY)
90       CONTINUE
C
C        INITIALIZE THE FRACTIONAL DISTANCES ON WEST AND EAST EDGES, SO
C        THAT GEOMETRY AT INTERIOR POINTS CAN BE CALCULATED
C
         DISTW(1) = 0.
         DISTE(1) = 0.
C
C        CALCULATE THE TOTAL DISTANCES ON WEST AND EAST EDGES
C
         DO 100 J = 2, NYRECT
            JM1      = J - 1
C
            INDJW    = 1 + (J  -1)*NXRECT
            INDJMW   = 1 + (JM1-1)*NXRECT
            INDJE    = J  *NXRECT
            INDJME   = JM1*NXRECT
C
            DXW      = GEOMG2(1,INDJW) - GEOMG2(1,INDJMW)
            DYW      = GEOMG2(2,INDJW) - GEOMG2(2,INDJMW)
            DXE      = GEOMG2(1,INDJE) - GEOMG2(1,INDJME)
            DYE      = GEOMG2(2,INDJE) - GEOMG2(2,INDJME)
C
            DISTW(J) = DISTW(JM1) + SQRT(DXW*DXW + DYW*DYW)
            DISTE(J) = DISTE(JM1) + SQRT(DXE*DXE + DYE*DYE)
C
100      CONTINUE

C        CALCULATE THE FRACTIONAL DISTANCES ON WEST AND EAST EDGES
C        FOR EACH NODE

         DO 110 J = 2, NYRECT
            DISTW(J) = DISTW(J)/DISTW(NYRECT)
            DISTE(J) = DISTE(J)/DISTE(NYRECT)
110      CONTINUE
C
C        STEP THROUGH EACH INTERIOR LINE AND SET GEOMETRY POINTS
C
         DO 130 I = 2, NXRECT-1
            FRACI = FLOAT(I-1)/FLOAT(NXRECT-1)
C
C           CALCULATE FRACTIONAL DISTANCES FOR EACH INTERIOR POINT
C
            DO 120 J = 2,NYRECT-1
               FRACJ         = (1.-FRACI)*DISTW(J) + FRACI*DISTE(J)
C
               IND           = I + (     J-1)*NXRECT
               INDN          = I + (NYRECT-1)*NXRECT
               INDS          = I
C
C              COMPUTE THE DISTANCE FROM NORTH EDGE TO SOUTH EDGE
C
```

```
            DELXNS        = GEOMG2(1,INDN) - GEOMG2(1,INDS)
           _ DELYNS        = GEOMG2(2,INDN) - GEOMG2(2,INDS)
C
C             COMPUTE LOCATION OF INTERIOR POINT
C
              GEOMG2(1,IND) = GEOMG2(1,INDS) + FRACJ*DELXNS
              GEOMG2(2,IND) = GEOMG2(2,INDS) + FRACJ*DELYNS
C
120         CONTINUE
130       CONTINUE
C
C         INITIALIZE THE SPECIFIC BOUNDARY CONDITION POINTERS
C         THIS SECTION CAN BE MODIFIED LATER FOR EDGES WITH
C         MULTIPLE BOUNDARY CONDITION TYPES

          NBNDG2 = 0
C
C         SOUTHWESTERN CORNER
C
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCSW
C
C         SOUTHERN EDGE
C
          DO 140 IBOUND = 2, NXRECT-1
            NBNDG2           = NBNDG2 + 1
            IBNDG2(5,NBNDG2) = IBCS
140       CONTINUE
C
C         SOUTHEASTERN CORNER
C
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCSE
C
C         EASTERN EDGE
C
          DO 150 IBOUND = 2, NYRECT-1
            NBNDG2           = NBNDG2 + 1
            IBNDG2(5,NBNDG2) = IBCE
150       CONTINUE
C
C         NORTHEASTERN CORNER
C
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCNE
C
C         NORTHERN EDGE
C
          DO 160 IBOUND = NXRECT-1, 2, -1
            NBNDG2           = NBNDG2 + 1
            IBNDG2(5,NBNDG2) = IBCN
160       CONTINUE
C
C         NORTHWESTERN CORNER
C
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCNW
```

```
C
C       WESTERN EDGE
C
        DO 170 IBOUND = NYRECT-1, 2, -1
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCW
170     CONTINUE

C       WRITE ALL THE GEOMETRY INFORMATION ON INPUTG.DAT SO THAT IT
C       CAN BE READ BY G2INIT LATER ON

        WRITE (INPUTG,180) NXRECT, NYRECT, NBNDG2, NNODG2
        WRITE (INPUTG,180) (IBNDG2(5,IB), IB=1,NBNDG2)
        WRITE (INPUTG,190) (GEOMG2(1,IN),GEOMG2(2,IN), IN=1,NNODG2)
180     FORMAT(12I5)
190     FORMAT(4G16.7)
C
C       NOW COMPUTE THE DEPENDENT VARIABLES
C       INITIALIZE VALUES FOR OXYGEN FOR LIGHTHILL DISSOCIATING GAS
C
        THETAD = 59500.
        RHOD   = 150.E03
        UGASFL = 8314.3
        AMWTA  = 16.0
        THETAD = THETAD/TREFFL
        RHOD   = RHOD/RHORFL
C
        ONEPA1 = 1. + ALPHAR
C       ONEPA1 IS 1 + ALPHAR

C       COMPUTE THE GRID AND OTHER QUANTITIES

        DO 220 I = 1, NNODG2
           XI    = GEOMG2(1,I)
           VCOMP = 0.
C
C          INLET STATION
C
           IF (XI .LE. XCONTA) THEN
              P     = PRESSI
              T     = TEMPI
              RHO   = RHOI
              ALPHA = ALPHAI
              UCOMP = UBEFOR
              IF (I .EQ. 1) GOTO 210
C             DO LINEAR INTERPOLATION FOR NON-EQUILIBRIUM LH-SHOCK,
C             IF NEED BE, OTHERWISE USE STEP FUNCTION INFORMATION
              IF (IGAS .EQ. 2) THEN
                DO 200 IP = 1, NPOINT-1
                  IF (XI .GE. X$(IP+1) .AND. XI .LE. X$(IP)) THEN
                     XRAT = (XI-X$(IP))/(X$(IP+1)-X$(IP))
                     DELTAA = A$(IP+1) - A$(IP)
                     DELTAP = P$(IP+1) - P$(IP)
                     DELTAR = R$(IP+1) - R$(IP)
                     DELTAT = T$(IP+1) - T$(IP)
                     DELTAU = U$(IP+1) - U$(IP)
                     ALPHA  = A$(IP) + DELTAA*XRAT
```

```
                  P      = P$(IP) + DELTAP*XRAT
     -            RHO    = R$(IP) + DELTAR*XRAT
                  T      = T$(IP) + DELTAT*XRAT
                  UCOMP  = U$(IP) + DELTAU*XRAT
                  GOTO 210
               ENDIF
200          CONTINUE
            ENDIF
C
C         EXIT STATION
C
            ELSE
               P     = PRESSE
               T     = TEMPE
               RHO   = RHOE
               ALPHA = ALPHAE
               UCOMP = UE
            ENDIF

210         V2    = UCOMP*UCOMP + VCOMP*VCOMP
            ETRHO = (TFACTR*T +ALPHA*THETAD)/ONEPA1 + V2/2.
            ET    = ETRHO*RHO

            DPENG2(1) = RHO
            DPENG2(2) = DPENG2(1)*UCOMP
            DPENG2(3) = DPENG2(1)*VCOMP
            DPENG2(4) = ET
            DPENG2(5) = DPENG2(1)*ALPHA
            WRITE (INPUTD,230) (DPENG2(K), K = 1, NTOTAL)
220      CONTINUE
230      FORMAT(8G15.7)
C
C        WRITE THE ADDITIONAL GRID INFORMATION ON AN UNFORMATTED FILE
C        FOR THE NON-EQUILIBRIUM LIGHTHILL-SHOCK, SO THAT IT CAN BE USED
C        BY A PRE-EMBEDDING ROUTINE
C
         IF (IGAS .EQ. 2) THEN
            OPEN (UNIT=58, FILE='LHSHOC.INT', STATUS='NEW',
     1           FORM='UNFORMATTED')
            WRITE(58) NPOINT, NTOTAL
            DO 240 IP = 1, NPOINT
               UCOMP     = U$(IP)
               V2        = UCOMP*UCOMP
               ETRHO     = (TFACTR*T$(IP)+A$(IP)*THETAD)/ONEPA1 +V2/2.
               DPENG2(1) = R$(IP)
               DPENG2(2) = R$(IP)*U$(IP)
               DPENG2(3) = 0.
               DPENG2(4) = ETRHO*R$(IP)
               DPENG2(5) = DPENG2(1)*A$(IP)
               WRITE (58) X$(IP),(DPENG2(K), K = 1, NTOTAL)
240         CONTINUE
         ENDIF

C        OPTION TO PLOT THE DATA

         WRITE(6,*) ' WANT TO PLOT DATA'
         READ (5,250) YESNO
```

```
250     FORMAT(A1)

        IF (YESNO .NE. 'Y' .AND. YESNO .NE. 'y') STOP ' THE END'

        CALL GR_INIT(5,6,MTITLE)
        REWIND (INPUTD)
        N$(1)       = NXRECT
        NLINE       = 1
        IOPT$(1)    = 2
        INDGR       = 21
        PLTITL(1:9) = 'DISTANCE'

        DO 260 I = 1, N$(1)
           X$(I) = GEOMG2(1,I)
           VCOMP = 0.
           READ (INPUTD,230) (DPENG2(K), K = 1, NTOTAL)
           RHO   = DPENG2(1)
           UCOMP = DPENG2(2)/DPENG2(1)
           ETRHO = DPENG2(4)/DPENG2(1)
           ALPHA = DPENG2(5)/DPENG2(1)
           V2    = UCOMP*UCOMP + VCOMP*VCOMP
           TDUM  = ETRHO - V2/2.
           TDUM  = TDUM*ONEPA1
           TDUM  = TDUM - ALPHA*THETAD
           T$(I) = TDUM/TFACTR
           A$(I) = ALPHA
           R$(I) = RHO
           U$(I) = UCOMP
           P$(I) = RHO*T$(I)*(1.+ALPHA)/ONEPA1
260     CONTINUE

270     WRITE(6,280)
280     FORMAT(1X,'THE FOLLOWING VARIABLES CAN BE PLOTTED VERSUS X'/
     1         5X,'1. DEGREE OF DISSOCIATION'/
     2         5X,'2. PRESSURE              '/
     3         5X,'3. DENSITY               '/
     4         5X,'4. TEMPERATURE           '/
     5         5X,'5. VELOCITY              '/
     6         5X,'6. EXIT                  '/ ' ===> ',$)

        READ (5,*) IPLOT
        PLTITL(10:22) = E1TAX(IPLOT)

        IF (IPLOT .EQ. 1) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,A$,N$)
        ELSE IF (IPLOT .EQ. 2) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,P$,N$)
        ELSE IF (IPLOT .EQ. 3) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,R$,N$)
        ELSE IF (IPLOT .EQ. 4) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,T$,N$)
        ELSE IF (IPLOT .EQ. 5) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,U$,N$)
        ENDIF

        WRITE (6,*) ' WANT TO PLOT MORE'
        READ (5,250) YESNO
```

```
      IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y') GOTO 270

C     EXAMPLE FILE : NISHOC.DAT
C  -1.                XMIN
C  2.                 XMAX
C  1.                 YMAX
C  -0.5               XCONTA
C  51                 NXRECT
C  11                 NYRECT
C  0.04               PERCEN
C  2                  IBCSW
C  3                  IBCS
C  1                  IBCSE
C  1                  IBCE
C  1                  IBCNE
C  3                  IBCN
C  2                  IBCNW
C  2                  IBCW

      STOP ' THE END'
      END
```

Bend duct

The program BEPIPE generates the initial grid for a circular curved duct with straight fore and aft ducts. The initial conditions are read from the output of a previous initial condition generator program.

```
      PROGRAM BEPIPE

      PARAMETER (MXX = 1000 ,   MYY = 500, MNODG2 = 5000)

      DIMENSION XSOUTH(MXX), XEAST(MYY), XNORTH(MXX), XWEST(MYY),
     1          YSOUTH(MXX), YEAST(MYY), YNORTH(MXX), YWEST(MYY),
     2          DISTW(MXX),  DISTE(MXX), GEOMG2(2,MNODG2),
     3          GEOP1(2,MNODG2), GEOP2(2,MNODG2), GEOP3(2,MNODG2),
     4          IBNDG2(5,500)  , DPENG2(5)

      DIMENSION A$(MXX), P$(MXX), R$(MXX), T$(MXX), U$(MXX), X$(MXX),
     1          IOPT$(1), N$(1) , E1TAX(6)
      CHARACTER MTITLE*80 , PLTITL*80 , E1TAX*12  , YESNO*1
      DATA E1TAX/ 'ALPHA      ', 'PRESSURE   ', 'DENSITY    ',
     1            'TEMPERATURE', 'VELOCITY   ', '           '/

C*********************************************************************
C
C     THIS PROGRAM GENERATES THE BOUNDARIES OF THE COMPUTATION DOMAIN
C     FOR A BEND PIPE WITH FORE AND AFT STRAIGHT PIPES INVOLVING A
C     MOVING SHOCK.  IT ALSO READS THE INITIAL CONDITIONS FOR THIS CASE
```

```
C       FROM FILE MOSHOC.IOT FOR A PERFECT OR A LIGHT-HILL GAS.  THIS FILE
C       MAY HAVE BEEN GENERATED BY EITHER MOSHOC.FOR OR LHSHOC.FOR.  THE
C       VARIABLES FOR THE CURRENT GEOMETRY ARE READ FROM THE FILE BEPIPE.DAT
C       THAT CONTAINS THE MIN/MAX RADII OF THE PIPE, LENGTHS OF THE FORE
C       AND AFT PIPES, OVERALL ANGLE OF THE CURVED PIPE, THE POSITION OF
C       INITIAL SHOCK AND THE NUMBER OF POINTS ALONG EACH SURFACE OF THE
C       PIPE.  THE BEND PIPE IS ASSUMED TO START AT X=0 AND Y BETWEEN
C       [RMIN,RMAX].   THE GRID IS GENERATED BY AN ALGEBRAIC CONSTRUCTION
C       BY SOLDERING THE THREE SECTIONS.  FINALLY THE PROGRAM INITIALIZES
C       THE DEPENDENT VARIABLES OVER ALL THE NODES.
C       THE OUTPUT FILE INPUTG.DAT CONTAINS GRID INFORMATION SUCH AS X
C       AND Y COORDINATES AT EACH NODE WHEREAS THE OUTPUT FILE INPUTD.DAT
C       CONTAINS DEPENDENT VARIABLE INFORMATION SUCH AT THESE NODES.
C
C       SUBROUTINES CALLED: ERRORM      UTILITY ROUTINE
C                           G2IBOG      STAR    ROUTINE
C                           GR_INIT     GRAFIC  ROUTINE
C                           GR_LINE     GRAFIC  ROUTINE
C
C***********************************************************************
C
C       INITIALIZATION
C
        INPIPE = 51
        INPUTG = 52
        INPUTD = 54
        JPRINT = 6
        IOPIPE = 8
        MOSHOC = 9
        PI     = 3.141592654
        RADIAN = PI/180.
        NPOINT = 0
        MTITLE = 'NON-EQUILIBRIUM  MOVING SHOCK'
C
C       OPEN THE APPROPRAITE UNITS
C
        OPEN (UNIT = INPIPE, FILE = 'BEPIPE.DAT', STATUS = 'OLD')
        OPEN (UNIT = INPUTG, FILE = 'INPUTG.DAT', STATUS = 'NEW')
        OPEN (UNIT = INPUTD, FILE = 'INPUTD.DAT', STATUS = 'NEW')
        OPEN (UNIT = IOPIPE, FILE = 'BEPIPE.OUT', STATUS = 'NEW')
        OPEN (UNIT = MOSHOC, FILE = 'MOSHOC.IOT', STATUS = 'OLD')
C
C       INPUT THE FOLLOWING QUANTITIES FROM FILE BEPIPE.DAT
C
C           XBEF      X-DISTANCE OF THE FORE PIPE
C           XAFT      X-DISTANCE OF THE AFT PIPE
C           RMIN      MINIMUM RADIUS OF THE PIPE
C           RMAX      MAXIMUM RADIUS OF THE PIPE
C           ANGLE     ANGLE (DEGREES) OF BEND PIPE
C           XCONTA    X-DISTANCE FOR THE SHOCK SURFACE
C           NXRECT1   NUMBER OF NODES ALONG X-DIRECTION IN FOREPIPE
C           NXRECT2   NUMBER OF NODES ALONG X-DIRECTION IN BENDPIPE
C           NXRECT3   NUMBER OF NODES ALONG X-DIRECTION IN AFTPIPE
C           NYRECT    NUMBER OF NODES ALONG Y-DIRECTION
C
        READ (INPIPE,*) XBEF
        READ (INPIPE,*) XAFT
```

```
      READ (INPIPE,*) RMIN
      READ-(INPIPE,*) RMAX
      READ (INPIPE,*) ANGLE
      READ (INPIPE,*) XCONTA
      READ (INPIPE,*) NXRECT1
      READ (INPIPE,*) NXRECT2
      READ (INPIPE,*) NXRECT3
      READ (INPIPE,*) NYRECT
C
C     READ THE FOLLOWING VALUES FROM MOSHOC.IOT
C
C         ALPHAI    INITIAL ALPHA AT INLET
C         ALPHAE    INITIAL ALPHA AT EXIT
C         ALPHAR    REFERENCE ALPHA
C         RHOI      INLET NON-DIMENSIONAL DENSITY
C         RHOE      EXIT NON-DIMENSIONAL DENSITY
C         UCOMPI    INLET NON-DIMENSIONAL VELOCITY
C         UCOMPE    EXIT NON-DIMENSIONAL VELOCITY
C         PRESSI    INLET NON-DIMENSIONAL PRESSURE
C         PRESSE    EXIT NON-DIMENSIONAL PRESSURE
C         TEMPI     INLET NON-DIMENSIONAL TEMPERATURE
C         TEMPE     EXIT NON-DIMENSIONAL TEMPERATURE
C         TREFFL    REFERENCE TEMPERATURE
C         SHKMAC    MACH NUMBER OF THE SHOCK
C         AGAS IGAS PARAMETER INDICATING TYPE OF GAS
C                   1: PERFECT     0: LIGHTHILL
C                   2: LIGHTHILL GAS FOR NON-EQUILIBRIUM
C
      READ (MOSHOC,10) ALPHAI
      READ (MOSHOC,10) ALPHAE
      READ (MOSHOC,10) ALPHAR
      READ (MOSHOC,10) RHOI
      READ (MOSHOC,10) RHOE
      READ (MOSHOC,10) UBEFOR
      READ (MOSHOC,10) UE
      READ (MOSHOC,10) PRESSI
      READ (MOSHOC,10) PRESSE
      READ (MOSHOC,10) TEMPI
      READ (MOSHOC,10) TEMPE
      READ (MOSHOC,10) TREFFL
      READ (MOSHOC,10) RHORFL
      READ (MOSHOC,10) SHKMAC
      READ (MOSHOC,10) AGAS
10    FORMAT (E15.8)
C
      IGAS = NINT(AGAS)
      NTOTAL = 5
      TFACTR = 3.
C
      IF (IGAS .EQ. 1) THEN
C        USE PERFECT GAS MODEL FOR THE SHOCK
C        READ SOME MORE VALUES FROM MOSHOC.IOT
         READ (MOSHOC,10) RGAS
         READ (MOSHOC,10) GAMMAI
         NTOTAL = 4
         ALPHAI = 0.
         ALPHAE = 0.
```

```
          ALPHAR = 0.
          TFACTR = 1./(GAMMAI-1.)
        ENDIF
C
        IF (IGAS .EQ. 2) THEN
C         USE LIGHTHILL GAS MODEL FOR THE NON-EQUILIBRIUM SHOCK
C         READ DEPENDENT VARIABLE VALUES FROM MOSHOC.IOT
          READ(MOSHOC,*) NPOINT
          DO 20 IP = 1, NPOINT
            READ(MOSHOC,30) X$(IP),A$(IP),P$(IP),R$(IP),T$(IP),U$(IP)
20        CONTINUE
        ENDIF
30      FORMAT(6E15.7)
C
C       WRITE DOWN THESE VALUES IN BEPIPE.OUT FOR THE PURPOSE OF KEEPING
C       A RECORD
C
        WRITE(IOPIPE,40) ALPHAI, ALPHAE, ALPHAR, RHOI, RHOE, TEMPI,
     1      TEMPE, PRESSI, PRESSI, TREFFL, RHORFL, XBEF, XAFT, RMIN, RMAX,
     2      ANGLE, XCONTA, NXRECT1, NXRECT2,NXRECT3,NYRECT, SHKMAC
40        FORMAT (' INITIAL ALPHA AT INLET             =',G14.5/
     1            ' INITIAL ALPHA AT EXIT              =',G14.5/
     1            ' REFERENCE ALPHA                    =',G14.5/
     1            ' RHOI                               =',G14.5/
     1            ' RHOE                               =',G14.5/
     1            ' TEMPI                              =',G14.5/
     1            ' TEMPE                              =',G14.5/
     1            ' PRESSI                             =',G14.5/
     1            ' PRESSE                             =',G14.5/
     1            ' TREFFL                             =',G14.5/
     1            ' RHORFL                             =',G14.5/
     1            ' LENGTH OF FORE PIPE                =',G14.5/
     1            ' LENGTH OF AFT PIPE                 =',G14.5/
     1            ' MINIMUM RADIUS OF BEND PIPE        =',G14.5/
     1            ' MAXIMUM RADIUS OF BEND PIPE        =',G14.5/
     1            ' ANGLE OF BEND PIPE                 =',G14.5/
     1            ' X-DISTANCE FOR THE CONTACT SURFACE =',G14.5/
     1            ' NUMBER OF X-NODES IN FOREPIPE      =',I5/
     1            ' NUMBER OF X-NODES IN BENDPIPE      =',I5/
     1            ' NUMBER OF X-NODES IN AFTPIPE       =',I5/
     1            ' NUMBER OF NODES ALONG Y-DIRECTION  =',I5/
     1            ' SHOCK MACK NUMBER                  =',G14.5/)

        NXRECT = NXRECT1 + NXRECT2 + NXRECT3 - 2

C       CHECK FOR OVERFLOW IN BOUNDARY NODE ARRAYS

        IF (NXREXT .GT. MXX) THEN
          ERR1 = NXRECT
          ERR2 = MXX
          CALL ERRORM (4,'G2RECT','NXRECT',ERR1,'MXX   ',ERR2,JPRINT,
     1      'NUMBER OF NODES EXCEEDS ITS LIMIT')
        ENDIF

        IF (NYREXT .GT. MYY) THEN
          ERR1 = NYRECT
          ERR2 = MYY
```

```
          CALL ERRORM (4,'G2RECT','NYRECT',ERR1,'MYY   ',ERR2,JPRINT,
     1      'NUMBER OF NODES EXCEEDS ITS LIMIT')
        ENDIF
C
C       SET THE BOUNDARY INFORMATION FOR FORE-PIPE
C       THE ORIGIN OF COORDINATES IS THE CENTER OF BEND PIPE CIRCLES
C       DELTA-Y IS CONSTANT FOR THE WHOLE PIPE ALTHOUGH DELTA-X MAY
C       VARY FROM ONE SECTION TO THE NEXT
C       NOW COMPUTE THE STEP SIZES
C
        XMAX   = 0.
        XMIN   = -XBEF
        YMIN   = RMIN
        YMAX   = RMAX
        DELX   = (XMAX-XMIN)/(NXRECT1-1)
        DELY   = (YMAX-YMIN)/(NYRECT-1)
C
C       COMPUTE (X,Y) COORDINATES FOR INFLOW/OUTFLOW BOUNDARIES
C
        DO 50 IY = 1, NYRECT
           XWEST(IY) = XMIN
           XEAST(IY) = XMAX
           YWEST(IY) = YMIN + (IY-1.)*DELY
           YEAST(IY) = YWEST(IY)
50      CONTINUE
C
C       COMPUTE (X,Y) COORDINATES FOR TOP/BOTTOM BOUNDARIES
C
        DO 60 IX = 1, NXRECT1
           XSOUTH(IX) = XMIN + (IX-1)*DELX
           XNORTH(IX) = XSOUTH(IX)
           YSOUTH(IX) = YMIN
           YNORTH(IX) = YMAX
60      CONTINUE
C
C       SET INTERIOR GRID FOR FORE-PIPE
C
        CALL G2IBOG (NXRECT1, NYRECT, XSOUTH, XEAST, XNORTH, XWEST,
     1               YSOUTH, YEAST, YNORTH, YWEST, GEOP1        )
C
C       PROCESS THE BEND PIPE COORDINATES

        ANGLE = ANGLE*RADIAN
        XMIN  = 0.
C
        DO 70 IY = 1, NYRECT
           XWEST(IY) = XMIN
           YWEST(IY) = YMIN + (IY-1.)*DELY
           XEAST(IY) = YWEST(IY)*SIN(ANGLE)
           YEAST(IY) = YWEST(IY)*COS(ANGLE)
70      CONTINUE
C
        DTHETA = ANGLE/(NXRECT2-1.)
        DO 80 IX = 1, NXRECT2
           THETA      = (IX-1.)*DTHETA
           XSOUTH(IX) = RMIN*SIN(THETA)
           XNORTH(IX) = RMAX*SIN(THETA)
```

```
          YSOUTH(IX) = RMIN*COS(THETA)
          YNORTH(IX) = RMAX*COS(THETA)
80      CONTINUE
C
C       SET INTERIOR GRID FOR BEND-PIPE
C
        CALL G2IBOG (NXRECT2, NYRECT, XSOUTH, XEAST, XNORTH, XWEST,
     1               YSOUTH, YEAST, YNORTH, YWEST, GEOP2        )

C       PROCESS THE AFT-PIPE COORDINATES

        DDL  = XAFT/(NXRECT3-1.)
C
        DO 90 IY = 1, NYRECT
          XWEST(IY) = XEAST(IY)
          YWEST(IY) = YEAST(IY)
          XEAST(IY) = XWEST(IY)+XAFT*COS(ANGLE)
          YEAST(IY) = YWEST(IY)-XAFT*SIN(ANGLE)
90      CONTINUE
C
        DO 100 IX = 1, NXRECT3
          DL         = (IX-1)*DDL
          XSOUTH(IX) =XWEST(1)+DL*COS(ANGLE)
          XNORTH(IX) =XWEST(NYRECT)+DL*COS(ANGLE)
          YSOUTH(IX) =YWEST(1)-DL*SIN(ANGLE)
          YNORTH(IX) =YWEST(NYRECT)-DL*SIN(ANGLE)
100     CONTINUE
C
C       SET INTERIOR GRID FOR AFT-PIPE
C
        CALL G2IBOG (NXRECT3, NYRECT, XSOUTH, XEAST, XNORTH, XWEST,
     1               YSOUTH , YEAST, YNORTH, YWEST, GEOP3        )
C
C       SOLDER THE THREE PIPES TOGETHER
C
        NNODG2 = 0
        NNODP1 = 0
        NNODP2 = 0
        NNODP3 = 0
C
        DO 140 IY = 1, NYRECT
C
C         PROCESS THE FIRST SECTION: INCLUDE ALL POINTS -- INTERIOR AND
C         EXTERIOR
C
          DO 110 IX = 1, NXRECT1
            NNODG2 = NNODG2 + 1
            NNODP1 = NNODP1 + 1
            GEOMG2(1,NNODG2) = GEOP1(1,NNODP1)
            GEOMG2(2,NNODG2) = GEOP1(2,NNODP1)
110       CONTINUE
C
C         PROCESS THE SECOND SECTION: INCLUDE ALL POINTS EXCEPT THE LEFT
C         OR INFLOW BOUNDARY
C
          NNODP2 = NNODP2 + 1
          DO 120 IX = 2, NXRECT2
```

```
            NNODG2 = NNODG2 + 1
            NNODP2 = NNODP2 + 1
            GEOMG2(1,NNODG2) = GEOP2(1,NNODP2)
            GEOMG2(2,NNODG2) = GEOP2(2,NNODP2)
120         CONTINUE
C
C           PROCESS THE THIRD SECTION: INCLUDE ALL POINTS EXCEPT THE LEFT
C           OR INFLOW BOUNDARY
C
            NNODP3 = NNODP3 + 1
            DO 130 IX = 2, NXRECT3
              NNODG2 = NNODG2 + 1
              NNODP3 = NNODP3 + 1
              GEOMG2(1,NNODG2) = GEOP3(1,NNODP3)
              GEOMG2(2,NNODG2) = GEOP3(2,NNODP3)
130         CONTINUE

140       CONTINUE
C
C         READ THE BOUNDARY CONDITION INDICATORS FROM BEPIPE.DAT
C         DIRECHLET : 2      REFLECTION : 1      WALL : 3
C
          READ (INPIPE,*) IBCSW
          READ (INPIPE,*) IBCS
          READ (INPIPE,*) IBCSE
          READ (INPIPE,*) IBCE
          READ (INPIPE,*) IBCNE
          READ (INPIPE,*) IBCN
          READ (INPIPE,*) IBCNW
          READ (INPIPE,*) IBCW
C
C         INITIALIZE THE SPECIFIC BOUNDARY CONDITION POINTERS
C         THIS SECTION CAN BE MODIFIED LATER FOR EDGES WITH
C         MULTIPLE BOUNDARY CONDITION TYPES
C
          NBNDG2 = 0
C
C         SOUTHWESTERN CORNER
C
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCSW
C
C         SOUTHERN EDGE
C
          DO 150 IBOUND = 2, NXRECT-1
            NBNDG2           = NBNDG2 + 1
            IBNDG2(5,NBNDG2) = IBCS
150       CONTINUE
C
C         SOUTHEASTERN CORNER
C
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCSE
C
C         EASTERN EDGE
C
          DO 160 IBOUND = 2, NYRECT-1
```

```
        NBNDG2           = NBNDG2 + 1
        IBNDG2(5,NBNDG2) = IBCE
160     CONTINUE
C
C       NORTHEASTERN CORNER
C
        NBNDG2           = NBNDG2 + 1
        IBNDG2(5,NBNDG2) = IBCNE
C
C       NORTHERN EDGE
C
        DO 170 IBOUND = NXRECT-1, 2, -1
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCN
170     CONTINUE
C
C       NORTHWESTERN CORNER
C
        NBNDG2           = NBNDG2 + 1
        IBNDG2(5,NBNDG2) = IBCNW
C
C       WESTERN EDGE
C
        DO 180 IBOUND = NYRECT-1, 2, -1
          NBNDG2           = NBNDG2 + 1
          IBNDG2(5,NBNDG2) = IBCW
180     CONTINUE
C
C       WRITE ALL THE GEOMETRY INFORMATION ON INPUTG.DAT SO THAT IT
C       CAN BE READ BY G2INIT LATER ON
C
        WRITE (INPUTG,190) NXRECT, NYRECT, NBNDG2, NNODG2
        WRITE (INPUTG,190) (IBNDG2(5,IB), IB=1,NBNDG2)
        WRITE (INPUTG,200) (GEOMG2(1,IN),GEOMG2(2,IN), IN=1,NNODG2)
190     FORMAT(12I5)
200     FORMAT(4G16.7)
C
C       NOW COMPUTE THE DEPENDENT VARIABLES
C       INITIALIZE VALUES FOR OXYGEN FOR LIGHTHILL DISSOCIATING GAS
C
        THETAD = 59500.
        RHOD   = 150.E03
        UGASFL = 8314.3
        AMWTA  = 16.0
        THETAD = THETAD/TREFFL
        RHOD   = RHOD/RHORFL
C
        ONEPA1 = 1. + ALPHAR
C       ONEPA1 IS 1 + ALPHAR

C       COMPUTE THE GRID AND OTHER QUANTITIES

        DO 230 I = 1, NNODG2
           XI    = GEOMG2(1,I)
           YI    = GEOMG2(2,I)
           VCOMP = 0.
C
```

```
C          INLET STATION
C
           IF (XI .LE. XCONTA .AND. YI .GE. RMIN) THEN
              P     = PRESSI
              T     = TEMPI
              RHO   = RHOI
              ALPHA = ALPHAI
              UCOMP = UBEFOR
              IF (I .EQ. 1) GOTO 220
C             DO LINEAR INTERPOLATION FOR NON-EQUILIBRIUM LH-SHOCK,
C             IF NEED BE, OTHERWISE USE STEP FUNCTION INFORMATION
              IF (IGAS .EQ. 2) THEN
                DO 210 IP = 1, NPOINT-1
                  IF (XI .GE. X$(IP+1) .AND. XI .LE. X$(IP)) THEN
                     XRAT = (XI-X$(IP))/(X$(IP+1)-X$(IP))
                     DELTAA = A$(IP+1) - A$(IP)
                     DELTAP = P$(IP+1) - P$(IP)
                     DELTAR = R$(IP+1) - R$(IP)
                     DELTAT = T$(IP+1) - T$(IP)
                     DELTAU = U$(IP+1) - U$(IP)
                     ALPHA  = A$(IP) + DELTAA*XRAT
                     P      = P$(IP) + DELTAP*XRAT
                     RHO    = R$(IP) + DELTAR*XRAT
                     T      = T$(IP) + DELTAT*XRAT
                     UCOMP  = U$(IP) + DELTAU*XRAT
                     GOTO 220
                  ENDIF
210             CONTINUE
              ENDIF
C
C          EXIT STATION
C
           ELSE
              P     = PRESSE
              T     = TEMPE
              RHO   = RHOE
              ALPHA = ALPHAE
              UCOMP = UE
           ENDIF

220        V2    = UCOMP*UCOMP + VCOMP*VCOMP
           ETRHO = (TFACTR*T +ALPHA*THETAD)/ONEPA1 + V2/2.
           ET    = ETRHO*RHO

           DPENG2(1) = RHO
           DPENG2(2) = DPENG2(1)*UCOMP
           DPENG2(3) = DPENG2(1)*VCOMP
           DPENG2(4) = ET
           DPENG2(5) = DPENG2(1)*ALPHA
           WRITE (INPUTD,240) (DPENG2(K), K = 1, NTOTAL)
230     CONTINUE
240     FORMAT(8G15.7)
C
C       WRITE THE ADDITIONAL GRID INFORMATION ON AN UNFORMATTED FILE
C       FOR THE NON-EQUILIBRIUM LIGHTHILL-SHOCK, SO THAT IT CAN BE USED
C       BY A PRE-EMBEDDING ROUTINE
C
```

```
      IF (IGAS .EQ. 2) THEN
         OPEN (UNIT=58, FILE='LHSHOC.INT', STATUS='NEW',
     1         FORM='UNFORMATTED')
         WRITE(58) NPOINT, NTOTAL
         DO 250 IP = 1, NPOINT
            UCOMP     = U$(IP)
            V2        = UCOMP*UCOMP
            ETRHO     = (TFACTR*T$(IP)+A$(IP)*THETAD)/ONEPA1 +V2/2.
            DPENG2(1) = R$(IP)
            DPENG2(2) = R$(IP)*U$(IP)
            DPENG2(3) = 0.
            DPENG2(4) = ETRHO*R$(IP)
            DPENG2(5) = DPENG2(1)*A$(IP)
            WRITE (58) X$(IP),(DPENG2(K), K = 1, NTOTAL)
250      CONTINUE
      ENDIF

C     OPTION TO PLOT THE DATA

      WRITE(6,*) ' WANT TO PLOT DATA'
      READ (5,260) YESNO
260   FORMAT(A1)

      IF (YESNO .NE. 'Y' .AND. YESNO .NE. 'y') STOP ' THE END'

      CALL GR_INIT(5,6,MTITLE)
      REWIND (INPUTD)
      N$(1)       = NXRECT
      NLINE       = 1
      IOPT$(1)    = 2
      INDGR       = 21
      PLTITL(1:9) = 'DISTANCE'

      DO 270 I = 1, N$(1)
         X$(I) = GEOMG2(1,I)
         VCOMP = 0.
         READ (INPUTD,240) (DPENG2(K), K = 1, NTOTAL)
         RHO   = DPENG2(1)
         UCOMP = DPENG2(2)/DPENG2(1)
         ETRHO = DPENG2(4)/DPENG2(1)
         ALPHA = DPENG2(5)/DPENG2(1)
         V2    = UCOMP*UCOMP + VCOMP*VCOMP
         TDUM  = ETRHO - V2/2.
         TDUM  = TDUM*ONEPA1
         TDUM  = TDUM - ALPHA*THETAD
         T$(I) = TDUM/TFACTR
         A$(I) = ALPHA
         R$(I) = RHO
         U$(I) = UCOMP
         P$(I) = RHO*T$(I)*(1.+ALPHA)/ONEPA1
270   CONTINUE

280   WRITE(6,290)
290   FORMAT(1X,'THE FOLLOWING VARIABLES CAN BE PLOTTED VERSUS X'/
     1       5X,'1. DEGREE OF DISSOCIATION'/
     2       5X,'2. PRESSURE              '/
     3       5X,'3. DENSITY               '/
```

```
     4         5X,'4. TEMPERATURE           '/
     5       - 5X,'5. VELOCITY              '/
     6         5X,'6. EXIT                  '/ ' ===> ',$)

        READ (5,*) IPLOT
        PLTITL(10:22) = E1TAX(IPLOT)

        IF (IPLOT .EQ. 1) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,A$,N$)
        ELSE IF (IPLOT .EQ. 2) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,P$,N$)
        ELSE IF (IPLOT .EQ. 3) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,R$,N$)
        ELSE IF (IPLOT .EQ. 4) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,T$,N$)
        ELSE IF (IPLOT .EQ. 5) THEN
          CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,X$,U$,N$)
        ENDIF

        WRITE (6,*) ' WANT TO PLOT MORE'
        READ (5,260) YESNO

        IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y') GOTO 280

C       EXAMPLE FILE : BEPIPE.DAT
C 1.                   XBEF
C 1.                   XAFT
C 0.5                  RMIN
C 1.0                  RMAX
C 90.0                 ANGLE
C -0.5                 XCONTA
C 11                   NXRECT1
C 11                   NXRECT3
C 11                   NXRECT3
C 6                    NYRECT
C 2                    IBCSW
C 3                    IBCS
C 1                    IBCSE
C 1                    IBCE
C 1                    IBCNE
C 3                    IBCN
C 2                    IBCNW
C 2                    IBCW

        STOP ' THE END'
        END
```

D.2 Block grid generator

This section contains information on the interactive block grid generator GNBLOC.

D.2.1 Common File

The file GNBLOC.INC includes declaration and common block statements. This file is to be included with the appropriate INCLUDE statements in the following FORTRAN code listing.

```
      PARAMETER (MBLOCK=20, MPOINT=100)
      COMMON/GENBLO/ NNODEH, NBLOCK, IBE, IBW, IBS, IBN, ISBLOCK,
     1               NXBLOCK, NYBLOCK, XSBLOCK, YSBLOCK,
     2               XNBLOCK, YNBLOCK, XEBLOCK, YEBLOCK,
     3               XWBLOCK, YWBLOCK
      DIMENSION IBE(MBLOCK,MPOINT), IBW(MBLOCK,MPOINT),
     1          IBS(MBLOCK,MPOINT), IBN(MBLOCK,MPOINT),
     2          ISBLOCK(MBLOCK,4) ,
     3          NXBLOCK(MBLOCK)   , NYBLOCK(MBLOCK),
     4          XSBLOCK(MBLOCK,MPOINT), YSBLOCK(MBLOCK,MPOINT),
     5          XNBLOCK(MBLOCK,MPOINT), YNBLOCK(MBLOCK,MPOINT),
     6          XEBLOCK(MBLOCK,MPOINT), YEBLOCK(MBLOCK,MPOINT),
     7          XWBLOCK(MBLOCK,MPOINT), YWBLOCK(MBLOCK,MPOINT)
C       This is a set of parameters for use with GR_GET_BIT and
C       GR_SET_BIT
C
        PARAMETER (IGR$CALCULATE_SCALES = 1)
        PARAMETER (IGR$DEPENDENT_SCALES = 2)
        PARAMETER (IGR$DRAW_AXES = 4)
        PARAMETER (IGR$DRAW_GRID = 8)
        PARAMETER (IGR$INTERACTIVE = 16)
        PARAMETER (IGR$NO_LOGO = 32)
        PARAMETER (IGR$NO_MENUS = 64)
        PARAMETER (IGR$BIT_CALCULATE_SCALES = 1)
        PARAMETER (IGR$BIT_DEPENDENT_SCALES = 2)
        PARAMETER (IGR$BIT_DRAW_AXES = 3)
        PARAMETER (IGR$BIT_DRAW_GRID = 4)
        PARAMETER (IGR$BIT_INTERACTIVE = 5)
        PARAMETER (IGR$BIT_NO_LOGO = 6)
        PARAMETER (IGR$BIT_NO_MENUS = 7)
C
C       parameters for GR_LINE
C
        PARAMETER (IGR$CLOSED_CURVE = 1)
        PARAMETER (IGR$PLOT_LINE = 2)
        PARAMETER (IGR$PLOT_SYMBOL = 4)
        PARAMETER (IGR$SMALL_SYMBOL = 8)
        PARAMETER (IGR$SYMBOL_FREQUENCY = 1024)
        PARAMETER (IGR$SYMBOL_TYPE = 65536)
```

```
PARAMETER (IGR$BIT_CLOSED_CURVE = 1)
PARAMETER (IGR$BIT_PLOT_LINE = 2)
PARAMETER (IGR$BIT_PLOT_SYMBOL = 3)
PARAMETER (IGR$BIT_SMALL_SYMBOL = 4)
```

D.2.2 Link information

The file GNBLOC.COM contains link information for the GNBLOC code.

```
$ LINK GNBLOC, GNSEPB, GNBNDG, GNREDN, ZRGNBN, GNDUMY,-
  GNDEBG, GNCONTR, GNGKIN, GNCHAN, GNLNOD, GNCLPO, GNWEDG, GNPINJ,-
  [PERVAIZ.STAR.OBJ]PSREDU, [PERVAIZ.STAR.OBJ]PSWRTU,-
  [PERVAIZ.STAR.OBJ]G2DIVU, [PERVAIZ.STAR.OBJ]G2CLPU,-
  [PERVAIZ.STAR.OBJ]G2BPIN, [PERVAIZ.STAR.OBJ]G2NODE,-
  [PERVAIZ.STAR.OBJ]A2CEWC, [PERVAIZ.PLT.OBJ]ZRPLTG,-
  [PERVAIZ.STAR.OBJ]M2AREA,-
  [PERVAIZ.ULT.OBJ]UL2LIB/LIB, [PERVAIZ.GRAFIC1]NEW_GRAFIC/LIB
```

D.2.3 Synopsis of variables

The file GNBLOC.DOC defines some variables in the GNBLOC code. For other variables consult the third section of this appendix.

```
------------------------------
SYNOPSIS OF VARIABLES IN GNBLOC
------------------------------

IBS (IB,IP)      THE ACTUAL NODE VALUE FOR THE POINT "IP" ON THE
                 SOUTHERN SURFACE OF BLOCK "IP"

                 . . . . . .

IBW (IB,IP)      THE ACTUAL NODE VALUE FOR THE POINT "IP" ON THE
                 WESTERN SURFACE OF BLOCK "IP"

ISBLOCK(IB,IP)   CONNECTIVITY ARRAY FOR JOINING SURFACES OF VARIOUS
                 BLOCKS.  "IB" INDICATES A BLOCK AMONG A TOTAL OF
                 "NBLOCK" BLOCKS.  "IS" INDICATES THE SIDE OR SURFACE
                 WITH THE FOLLOWING MEANING:
                        1: SOUTH      2: EAST
                        3: NORTH      4: WEST
                 THE VALUE OF THE ARRAY HAS THE FOLLOWING MEANING:
                   >0 SURFACE IS NOT ON A PHYSICAL BOUNDARY
                   <0 SURFACE IS ON A PHYSICAL BOUNDARY AND INDICATES
                      BOUNDARY CONDITION TYPE.
                   POSITIVE VALUE INDICATES THAT THE BOUNDARY POINTS
                   HAVE BEEN INCLUDED IN THE BLOCK WITH THAT VALUE AND
                   HENCE MUST NOT BE INCLUDED HERE.  A ZERO VALUE
```

```
                    INDICATES THAT THE BOUNDARY POINTS MUST BE INCLUDED
                    IN THE CURRENT BLOCK.

NBLOCK              TOTAL NUMBER OF BLOCKS
NNODEH              CURRENT TOTAL NUMBER OF NODES
NXBLOCK(IB)         NUMBER OF HORIZONTAL POINTS IN BLOCK IB
NYBLOCK(IB)         NUMBER OF VERTICAL POINTS IN BLOCK IB

XSBLOCK(IB,IP)      X-COORDINATE OF THE POINT "IP" ON THE SOUTHERN
                    BOUNDARY IN BLOCK "IB"
YSBLOCK(IB,IP)      Y-COORDINATE OF THE POINT "IP" ON THE SOUTHERN
                    BOUNDARY IN BLOCK "IB"

                    . . . . . .

XWBLOCK(IB,IP)      X-COORDINATE OF THE POINT "IP" ON THE WESTERN
                    BOUNDARY IN BLOCK "IB"
YWBLOCK(IB,IP)      Y-COORDINATE OF THE POINT "IP" ON THE WESTERN
                    BOUNDARY IN BLOCK "IB"


FOR OTHER VARIABLES AND COMMON BLOCK REFER TO THE FILE
STAR.DOC IN THE STAR CODE LISTING.
```

D.2.4 Input to GNBLOC

The code GNBLOC requires the specification of the boundary points and coordinates of each block in the grid structure. It also requires the specification of the connectivity of the individual blocks in the overall assembly. As an example, a listing of a program STRUT is provided here that generates this information. Figure (D.2) shows the blocks, the number of points on each surface and the connectivity of blocks.

```
      PROGRAM STRUT

      PARAMETER (MBLOCK=20, MPOINT=100)

      DIMENSION XCOR(MBLOCK,4), YCOR(MBLOCK,4), ISBLOCK(MBLOCK,4),
     1          NXBLOCK(MBLOCK)   , NYBLOCK(MBLOCK),
     2          XSBLOCK(MBLOCK,MPOINT), YSBLOCK(MBLOCK,MPOINT),
     3          XNBLOCK(MBLOCK,MPOINT), YNBLOCK(MBLOCK,MPOINT),
     4          XEBLOCK(MBLOCK,MPOINT), YEBLOCK(MBLOCK,MPOINT),
     5          XWBLOCK(MBLOCK,MPOINT), YWBLOCK(MBLOCK,MPOINT)

C*********************************************************************

C       THIS PROGRAM GENERATES THE BOUNDARIES OF VARIOUS BLOCK FOR A
C       SCRAM-JET CASE GRID IN THE A. KUMAR PAPER.  THE OUTPUT WRITTEN
C       BY THIS PROGRAM IS PROCESSED BY GENBLC.
```

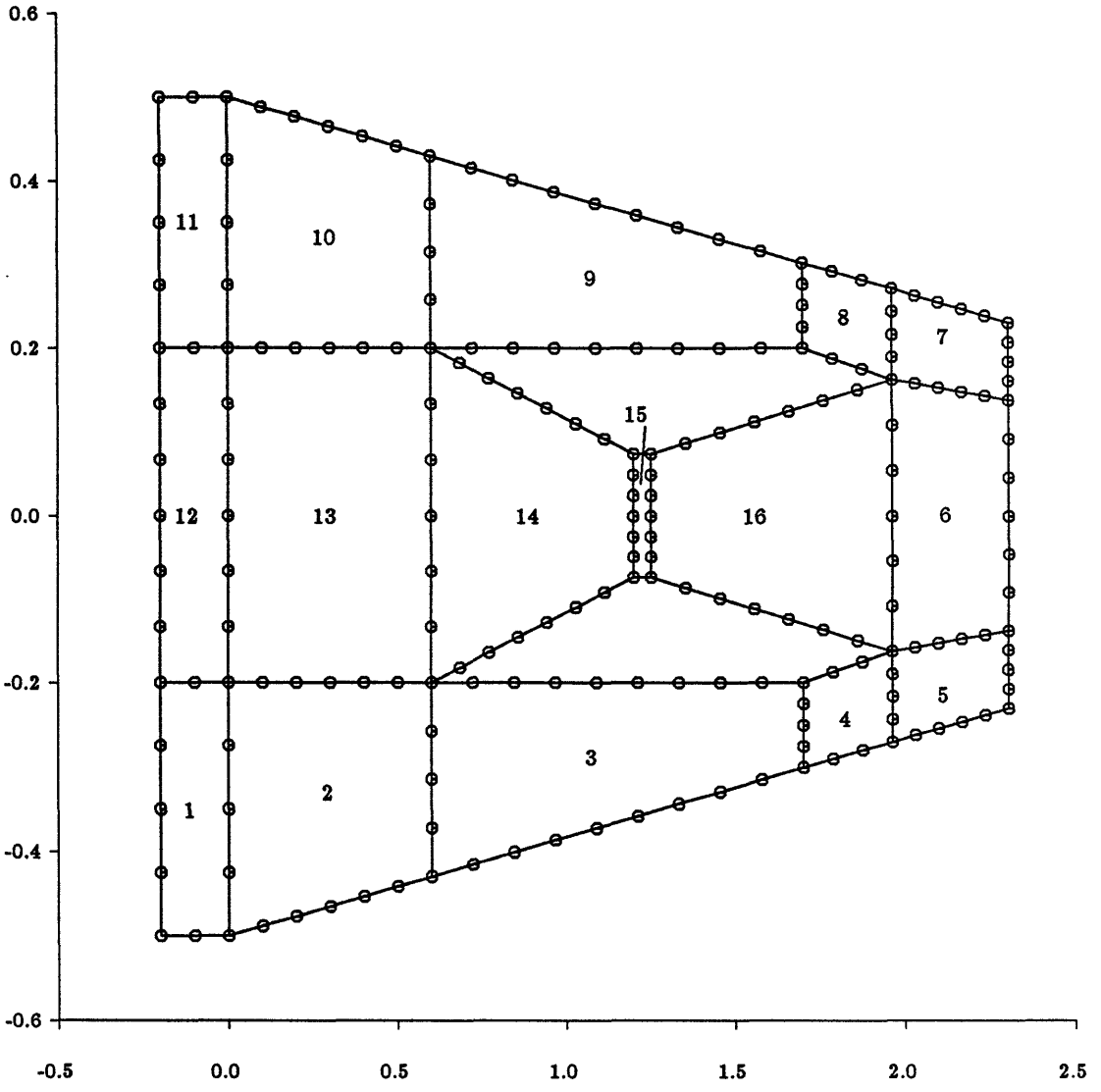

Figure D.2: Blocks and their connectivity for a two strut geometry.

```
C*********************************************************************
C
      IGENBC = 51
      PI     = 3.1415926
      RADIAN = PI/180.

      OPEN (UNIT = IGENBC, FILE = 'GNBINP.DAT', STATUS = 'NEW')
C
C     SETUP THE DETAILS OF EACH BLOCK
C
      ALPHA      = 6.668*RADIAN
      TANALP     = TAN(ALPHA)
      BETA       = 11.873*RADIAN
c     BETA       = 16.*RADIAN
      TANBET     = TAN(BETA)

C     BLOCK 1
      XCOR(1,1)  =-0.2
      XCOR(1,2)  = 0.
      XCOR(1,3)  = 0.
      XCOR(1,4)  =-0.2
      YCOR(1,1)  =-0.5
      YCOR(1,2)  =-0.5
      YCOR(1,3)  =-0.2
      YCOR(1,4)  =-0.2
      NXBLOCK(1) = 3
      NYBLOCK(1) = 5
      ISBLOCK(1,1) =-3
      ISBLOCK(1,2) = 0
      ISBLOCK(1,3) = 0
      ISBLOCK(1,4) =-2

C     BLOCK 2
      XCOR(2,1)  = 0.
      XCOR(2,2)  = 0.6
      XCOR(2,3)  = 0.6
      XCOR(2,4)  = 0.
      YCOR(2,1)  =-0.5
      YCOR(2,2)  = YCOR(1,1) + XCOR(2,2)*TANALP
      YCOR(2,3)  =-0.2
      YCOR(2,4)  =-0.2
      NXBLOCK(2) = 7
      NYBLOCK(2) = NYBLOCK(1)
      ISBLOCK(2,1) =-3
      ISBLOCK(2,2) = 0
      ISBLOCK(2,3) = 0
      ISBLOCK(2,4) = 1

C     BLOCK 3
      XCOR(3,1)  = XCOR(2,2)
      XCOR(3,2)  = 1.7
      XCOR(3,3)  = XCOR(3,2)
      XCOR(3,4)  = XCOR(2,3)
      YCOR(3,1)  = YCOR(2,2)
      YCOR(3,2)  = YCOR(2,1) + XCOR(3,2)*TANALP
      YCOR(3,4)  = YCOR(2,3)
```

```
      YCOR(3,3)  = YCOR(3,4)
      NXBLOCK(3) = 10
      NYBLOCK(3) = NYBLOCK(1)
      ISBLOCK(3,1) =-3
      ISBLOCK(3,2) = 0
      ISBLOCK(3,3) =-3
      ISBLOCK(3,4) = 2

C     BLOCK 4
      XCOR(4,1)  = XCOR(3,2)
      XCOR(4,2)  = XCOR(4,1) + 0.262
      XCOR(4,3)  = XCOR(4,2)
      XCOR(4,4)  = XCOR(3,3)
      YCOR(4,1)  = YCOR(3,2)
      YCOR(4,2)  = YCOR(2,1) + XCOR(4,2)*TANALP
      YCOR(4,3)  = -0.1625
      YCOR(4,4)  = YCOR(3,3)
      NXBLOCK(4) = 4
      NYBLOCK(4) = NYBLOCK(1)
      ISBLOCK(4,1) =-3
      ISBLOCK(4,2) = 0
      ISBLOCK(4,3) =-3
      ISBLOCK(4,4) = 3

C     BLOCK 5
      XCOR(5,1)  = XCOR(4,2)
      XCOR(5,2)  = 2.3063
      XCOR(5,3)  = XCOR(5,2)
      XCOR(5,4)  = XCOR(4,3)
      YCOR(5,1)  = YCOR(4,2)
      YCOR(5,2)  = YCOR(2,1) + XCOR(5,2)*TANALP
      YCOR(5,4)  = YCOR(4,3)
      YNEXT      = YCOR(4,4) + (YCOR(4,3)-YCOR(4,4))*
     1               (XCOR(5,2)-XCOR(4,4))/(XCOR(4,3)-XCOR(4,4))
      YNEXT      = 0.5*(YNEXT+YCOR(4,3))
      YCOR(5,3)  = YNEXT
      NXBLOCK(5) = 6
      NYBLOCK(5) = NYBLOCK(1)
      ISBLOCK(5,1) =-3
      ISBLOCK(5,2) =-1
      ISBLOCK(5,3) = 0
      ISBLOCK(5,4) = 4

C     BLOCK 6
      XCOR(6,1)  = XCOR(5,4)
      XCOR(6,2)  = XCOR(5,3)
      XCOR(6,3)  = XCOR(6,2)
      XCOR(6,4)  = XCOR(6,1)
      YCOR(6,1)  = YCOR(5,4)
      YCOR(6,2)  = YCOR(5,3)
      YCOR(6,3)  =-YCOR(6,2)
      YCOR(6,4)  =-YCOR(6,1)
      NXBLOCK(6) = NXBLOCK(5)
      NYBLOCK(6) = 7
      ISBLOCK(6,1) = 5
      ISBLOCK(6,2) =-1
      ISBLOCK(6,3) = 0
```

```
      ISBLOCK(6,4) = 0

C     BLOCK 7
      XCOR(7,1)  = XCOR(5,4)
      XCOR(7,2)  = XCOR(5,3)
      XCOR(7,3)  = XCOR(5,2)
      XCOR(7,4)  = XCOR(5,1)
      YCOR(7,1)  =-YCOR(5,4)
      YCOR(7,2)  =-YCOR(5,3)
      YCOR(7,3)  =-YCOR(5,2)
      YCOR(7,4)  =-YCOR(5,1)
      NXBLOCK(7) = NXBLOCK(5)
      NYBLOCK(7) = NYBLOCK(5)
      ISBLOCK(7,1) = 6
      ISBLOCK(7,2) =-1
      ISBLOCK(7,3) =-3
      ISBLOCK(7,4) = 0

C     BLOCK 8
      XCOR(8,1)  = XCOR(4,4)
      XCOR(8,2)  = XCOR(4,3)
      XCOR(8,3)  = XCOR(4,2)
      XCOR(8,4)  = XCOR(4,1)
      YCOR(8,1)  =-YCOR(4,4)
      YCOR(8,2)  =-YCOR(4,3)
      YCOR(8,3)  =-YCOR(4,2)
      YCOR(8,4)  =-YCOR(4,1)
      NXBLOCK(8) = NXBLOCK(4)
      NYBLOCK(8) = NYBLOCK(4)
      ISBLOCK(8,1) =-3
      ISBLOCK(8,2) = 7
      ISBLOCK(8,3) =-3
      ISBLOCK(8,4) = 0

C     BLOCK 9
      XCOR(9,1)  = XCOR(3,4)
      XCOR(9,2)  = XCOR(3,3)
      XCOR(9,3)  = XCOR(3,2)
      XCOR(9,4)  = XCOR(3,1)
      YCOR(9,1)  =-YCOR(3,4)
      YCOR(9,2)  =-YCOR(3,3)
      YCOR(9,3)  =-YCOR(3,2)
      YCOR(9,4)  =-YCOR(3,1)
      NXBLOCK(9) = NXBLOCK(3)
      NYBLOCK(9) = NYBLOCK(3)
      ISBLOCK(9,1) =-3
      ISBLOCK(9,2) = 8
      ISBLOCK(9,3) =-3
      ISBLOCK(9,4) = 0

C     BLOCK 10
      XCOR(10,1)  = XCOR(2,4)
      XCOR(10,2)  = XCOR(2,3)
      XCOR(10,3)  = XCOR(2,2)
      XCOR(10,4)  = XCOR(2,1)
      YCOR(10,1)  =-YCOR(2,4)
      YCOR(10,2)  =-YCOR(2,3)
```

```
      YCOR(10,3)  =-YCOR(2,2)
      YCOR(10,4)  =-YCOR(2,1)
      NXBLOCK(10) = NXBLOCK(2)
      NYBLOCK(10) = NYBLOCK(2)
      ISBLOCK(10,1) = 0
      ISBLOCK(10,2) = 9
      ISBLOCK(10,3) =-3
      ISBLOCK(10,4) = 0

C     BLOCK 11
      XCOR(11,1)  = XCOR(1,4)
      XCOR(11,2)  = XCOR(1,3)
      XCOR(11,3)  = XCOR(1,2)
      XCOR(11,4)  = XCOR(1,1)
      YCOR(11,1)  =-YCOR(1,4)
      YCOR(11,2)  =-YCOR(1,3)
      YCOR(11,3)  =-YCOR(1,2)
      YCOR(11,4)  =-YCOR(1,1)
      NXBLOCK(11) = NXBLOCK(1)
      NYBLOCK(11) = NYBLOCK(1)
      ISBLOCK(11,1) = 0
      ISBLOCK(11,2) = 10
      ISBLOCK(11,3) =-3
      ISBLOCK(11,4) =-2

C     BLOCK 12
      XCOR(12,1)  = XCOR(1,4)
      XCOR(12,2)  = XCOR(1,3)
      XCOR(12,3)  = XCOR(11,2)
      XCOR(12,4)  = XCOR(11,1)
      YCOR(12,1)  = YCOR(1,4)
      YCOR(12,2)  = YCOR(1,3)
      YCOR(12,3)  = YCOR(11,2)
      YCOR(12,4)  = YCOR(11,1)
      NXBLOCK(12) = NXBLOCK(1)
      NYBLOCK(12) = NYBLOCK(6)
      ISBLOCK(12,1) = 1
      ISBLOCK(12,2) = 0
      ISBLOCK(12,3) = 11
      ISBLOCK(12,4) =-2

C     BLOCK 13
      XCOR(13,1)  = XCOR(2,4)
      XCOR(13,2)  = XCOR(2,3)
      XCOR(13,3)  = XCOR(10,2)
      XCOR(13,4)  = XCOR(10,1)
      YCOR(13,1)  = YCOR(2,4)
      YCOR(13,2)  = YCOR(2,3)
      YCOR(13,3)  = YCOR(10,2)
      YCOR(13,4)  = YCOR(10,1)
      NXBLOCK(13) = NXBLOCK(2)
      NYBLOCK(13) = NYBLOCK(6)
      ISBLOCK(13,1) = 2
      ISBLOCK(13,2) = 0
      ISBLOCK(13,3) = 10
      ISBLOCK(13,4) =12
```

```
C       BLOCK 14
        XCOR(14,1)  = XCOR(3,4)
        XCOR(14,2)  = 1.2
        XCOR(14,3)  = XCOR(14,2)
        XCOR(14,4)  = XCOR(14,1)
        YCOR(14,1)  = YCOR(3,4)
        YCOR(14,2)  = YCOR(14,1) + TANBET*(XCOR(14,2)-XCOR(14,1))
        YCOR(14,3)  =-YCOR(14,2)
        YCOR(14,4)  =-YCOR(14,1)
        NXBLOCK(14) = 8
        NYBLOCK(14) = NYBLOCK(6)
        ISBLOCK(14,1) =-3
        ISBLOCK(14,2) = 0
        ISBLOCK(14,3) =-3
        ISBLOCK(14,4) =13

C       BLOCK 15
        XCOR(15,1)  = XCOR(14,2)
        XCOR(15,2)  = XCOR(15,1) + 0.0524
        XCOR(15,3)  = XCOR(15,2)
        XCOR(15,4)  = XCOR(15,1)
        YCOR(15,1)  = YCOR(14,2)
        YCOR(15,2)  = YCOR(15,1)
        YCOR(15,3)  =-YCOR(15,2)
        YCOR(15,4)  =-YCOR(15,1)
        NXBLOCK(15) = 2
        NYBLOCK(15) = NYBLOCK(13)
        ISBLOCK(15,1) =-3
        ISBLOCK(15,2) = 0
        ISBLOCK(15,3) =-3
        ISBLOCK(15,4) =14

C       BLOCK 16
        XCOR(16,1)  = XCOR(15,2)
        XCOR(16,2)  = XCOR(4,3)
        XCOR(16,3)  = XCOR(7,1)
        XCOR(16,4)  = XCOR(15,3)
        YCOR(16,1)  = YCOR(15,2)
        YCOR(16,2)  = YCOR(4,3)
        YCOR(16,3)  = YCOR(7,1)
        YCOR(16,4)  = YCOR(15,3)
        NXBLOCK(16) = 8
        NYBLOCK(16) = NYBLOCK(13)
        ISBLOCK(16,1) =-3
        ISBLOCK(16,2) = 6
        ISBLOCK(16,3) =-3
        ISBLOCK(16,4) =15
C
C       INPUT THE TOTAL NUMBER OF BLOCKS, NBLOCK, TO BE LINKED
C
        NBLOCK = 16
        WRITE (IGENBC,*) NBLOCK

        DO 10 IBLOCK = 1, NBLOCK
C          SOUTHERN BOUNDARY
           DX = (XCOR(IBLOCK,2)-XCOR(IBLOCK,1))/(NXBLOCK(IBLOCK)-1.)
           DY = (YCOR(IBLOCK,2)-YCOR(IBLOCK,1))/(NXBLOCK(IBLOCK)-1.)
```

```
            DO IX = 1, NXBLOCK(IBLOCK)
              XSBLOCK(IBLOCK,IX) = XCOR(IBLOCK,1) + DX*(IX-1)
              YSBLOCK(IBLOCK,IX) = YCOR(IBLOCK,1) + DY*(IX-1)
            ENDDO
C           EASTERN BOUNDARY
            DX = (XCOR(IBLOCK,3)-XCOR(IBLOCK,2))/(NYBLOCK(IBLOCK)-1.)
            DY = (YCOR(IBLOCK,3)-YCOR(IBLOCK,2))/(NYBLOCK(IBLOCK)-1.)
            DO IX = 1, NYBLOCK(IBLOCK)
               XEBLOCK(IBLOCK,IX) = XCOR(IBLOCK,2) + DX*(IX-1)
               YEBLOCK(IBLOCK,IX) = YCOR(IBLOCK,2) + DY*(IX-1)
            ENDDO

C           NORTHERN BOUNDARY
            DX = (XCOR(IBLOCK,3)-XCOR(IBLOCK,4))/(NXBLOCK(IBLOCK)-1.)
            DY = (YCOR(IBLOCK,3)-YCOR(IBLOCK,4))/(NXBLOCK(IBLOCK)-1.)
            DO IX = 1, NXBLOCK(IBLOCK)
               XNBLOCK(IBLOCK,IX) = XCOR(IBLOCK,4) + DX*(IX-1)
               YNBLOCK(IBLOCK,IX) = YCOR(IBLOCK,4) + DY*(IX-1)
            ENDDO
C           WESTERN BOUNDARY
            DX = (XCOR(IBLOCK,4)-XCOR(IBLOCK,1))/(NYBLOCK(IBLOCK)-1.)
            DY = (YCOR(IBLOCK,4)-YCOR(IBLOCK,1))/(NYBLOCK(IBLOCK)-1.)
            DO IX = 1, NYBLOCK(IBLOCK)
               XWBLOCK(IBLOCK,IX) = XCOR(IBLOCK,1) + DX*(IX-1)
               YWBLOCK(IBLOCK,IX) = YCOR(IBLOCK,1) + DY*(IX-1)
            ENDDO
10        CONTINUE
C
          DO 50 IBLOCK = 1, NBLOCK
C
C           INPUT THE NUMBER OF HORIZONTAL AND VERTICAL POINTS IN EACH BLOCK
C
            WRITE (IGENBC,*) NXBLOCK(IBLOCK), NYBLOCK(IBLOCK)
C
C           WRITE THE GEOMETRY OF SOUTHERN AND NORTHERN BOUNDARY
C
            DO 30 IX = 1, NXBLOCK(IBLOCK)
            WRITE (IGENBC,*) XSBLOCK(IBLOCK,IX), YSBLOCK(IBLOCK,IX),
     1                       XNBLOCK(IBLOCK,IX), YNBLOCK(IBLOCK,IX)
30          CONTINUE
C
C           WRITE THE GEOMETRY OF EASTERN AND WESTERN BOUNDARY
C
            DO 40 IY = 1, NYBLOCK(IBLOCK)
            WRITE (IGENBC,*) XEBLOCK(IBLOCK,IY), YEBLOCK(IBLOCK,IY),
     1                       XWBLOCK(IBLOCK,IY), YWBLOCK(IBLOCK,IY)
40          CONTINUE
C
C           INPUT THE CONNECTIVITY ARRAY FOR JOINING SURFACES OF
C           VARIOUS BLOCKS, 0 OR >0 MEANS THAT THE SURFACE IS NOT
C           ON A PHYSICAL BOUNDARY,  WHEREAS <0 MEANS THAT IT IS
C           ON A PHYSICAL BOUNDARY (B.C. TYPE IS USED AS NEGATIVE).
C           POSITIVE VALUE INDICATES THE ADJASCENT BLOCK, WHEREAS
C           ZERO MEANS THE ADJACENT BLOCK DOES NOT MATTER
C
            WRITE (IGENBC,*) (ISBLOCK(IBLOCK,JS), JS=1,4)
C
```

```
50      CONTINUE
C
C       SPECIAL CORNER BOUNDARY CONDITIONS
C       THE NEGATIVE VALUES INDICATE THAT NO CHANGE IS DESIRED
C       AT THE CORNER BOUNDARY CONDITIONS
        IBCSW = 2
        IBCSE = 3
        IBCNE = 3
        IBCNW = 2

        WRITE (IGENBC,*) IBCSW, IBCSE, IBCNE, IBCNW

        END
```

D.2.5 Listing of GNBLOC code

Main routine

```
        PROGRAM GNBLOC

        INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] A2COMN.INC /LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] CHCOMN.INC /LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] E2COMN.INC /LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] FLCOMN.INC /LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] IOCOMN.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] KYCOMN.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] PRCOMN.INC /LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] TICOMN.INC /LIST'
        INCLUDE 'GNBLOC.INC/LIST'
        REAL*4    GRDUMY(30), ALIMITS(6), XMIN,XMAX,YMIN,YMAX
        DIMENSION ZX(MNODG2), ZY(MNODG2)

        CHARACTER PLTITL*96
        EXTERNAL  ZRGNBN, ZRPLTG

C*********************************************************************

C       THIS PROGRAM GENERATES A GRID BY SOLDERING VARIOUS BLOCKS OF
C       SUB-GRIDS.  EACH SUB-GRID IS GENERATED BY AN ALGEBRAIC METHOD.

C*********************************************************************
C
C       INITIALIZE POINTERS FOR ALL LEVELS
C
        DO 10 ILEVEL = -MLVLG2, MLVLG2
           ILVLG2(1,ILEVEL) = 0
           ILVLG2(2,ILEVEL) = 0
           ILVLG2(3,ILEVEL) = 0
10      CONTINUE
```

```
C
C       INITIALIZE THE NEIGHBOUR CELL ARRAY
C          -
        DO 20 K = 1, 4
          DO 20 KN = 1, MNODG2
           NEIBG2(K,KN) = 0
20      CONTINUE

        JPNTRE = 51
        JGIVEN = 23
        JPRINT = 6
        JTERMI = 5
        JTERMO = 6
C
        MTITLE   = ' '
        WRITE(PLTITL,25)
25      FORMAT(' X-AXIS  Y-AXIS   GRID PLOT ')
        CALL GR_INIT(JTERMI, JTERMO, MTITLE)

        WRITE(JTERMO,30)
30      FORMAT(5X,'INPUT IFROMU TO INDICATE PLACE FROM WHERE THE',
     1         1X,'GRID INFORMATION IS TO BE READ'/
     2        10X,'1.  BLOCK BOUNDARY INPUT DATA (GNBINP.DAT) '/
     3        10X,'2.  BLOCK OUTPUT DATA         (GNBLOC.DAT) '/
     4        10X,'3.  ACTUAL PREVIOUS RUN DATA  (JPNTRE.DAT) '/
     5        10X,'===> ',$)
        READ (JTERMI,*) IFROMU
        GOTO (40,1100,2100), IFROMU
        STOP ' THE END'
C
40      OPEN (UNIT = JPNTRE, FILE = 'GNBINP.DAT', STATUS = 'OLD')
        OPEN (UNIT = JGIVEN, FILE = 'GNBINP.OUT', STATUS = 'NEW')
C
C       INPUT THE TOTAL NUMBER OF BLOCKS, NBLOCK, TO BE LINKED
C
        READ (JPNTRE,*) NBLOCK

        DO 70 IBLOCK = 1, NBLOCK
C
C         INPUT THE NUMBER OF HORIZONTAL AND VERTICAL POINTS IN EACH BLOCK
C
          READ (JPNTRE,*) NXBLOCK(IBLOCK), NYBLOCK(IBLOCK)
C
C         READ THE GEOMETRY OF SOUTHERN AND NORTHERN BOUNDARY
C
          DO 50 IX = 1, NXBLOCK(IBLOCK)
          READ (JPNTRE,*) XSBLOCK(IBLOCK,IX), YSBLOCK(IBLOCK,IX),
     1                    XNBLOCK(IBLOCK,IX), YNBLOCK(IBLOCK,IX)
50        CONTINUE
C
C         READ THE GEOMETRY OF EASTERN AND WESTERN BOUNDARY
C
          DO 60 IY = 1, NYBLOCK(IBLOCK)
          READ (JPNTRE,*) XEBLOCK(IBLOCK,IY), YEBLOCK(IBLOCK,IY),
     1                    XWBLOCK(IBLOCK,IY), YWBLOCK(IBLOCK,IY)
60        CONTINUE
C
```

```
C         INPUT THE CONNECTIVITY ARRAY FOR JOINING SURFACES OF
C         VARIOUS BLOCKS, 0 OR >0 MEANS THAT THE SURFACE IS NOT
C         ON A PHYSICAL BOUNDARY,  WHEREAS <0 MEANS THAT IT IS
C         ON A PHYSICAL BOUNDARY (B.C. TYPE IS USED AS NEGATIVE).
C         POSITIVE VALUE INDICATES THE ADJASCENT BLOCK, WHEREAS
C         ZERO MEANS THE ADJACENT BLOCK DOES NOT MATTER
C
          READ (JPNTRE,*) (ISBLOCK(IBLOCK,JS), JS=1,4)
C
70        CONTINUE
C
C         READ THE SPECIAL BOUNDARY CONDITION POINTERS AT CORNERS
          READ (JPNTRE,*) IBCSW, IBCSE, IBCNE, IBCNW

          WRITE(JTERMO,80)
80        FORMAT(5X,'INPUT INDGR IN A BINARY-CODED MANNER '/
     1         10X,'-1     NEGATIVE TO SKIP GRAPH'/
     2         10X,' 0     PLOT FULL DATA'/
     3         10X,' 1     AUTOMATIC SCALE CALCULATION'/
     4         10X,' 2     INPUT YOUR OWN SCALES'/
     5         10X,' 4     DRAW AXES'/
     6         10X,' 8     DRAW BACKGROUD GRID'/
     7         10X,' 16    PLOT ON TERMINAL'/
     8         10X,' >100  NO SYMBOLS'/
     8         10X,'===> ',$)
          READ (JTERMI,*) INDGR
          A1 = -2
          IF (INDGR .GE. 100) THEN
             INDGR = INDGR - 100
             A1    = 0
          ENDIF
          IF (INDGR .LT. 0) GOTO 100
          IF (INDGR .NE. 0) GOTO 96

          INDGR = 22
          XMIN  = 1000.
          XMAX  =-1000.
          YMIN  = 1000.
          YMAX  =-1000.

          DO 90 IBLOCK = 1, NBLOCK
             DO 93 IY = 1, NYBLOCK(IBLOCK)
                YMIN = MIN (YMIN,YSBLOCK(IBLOCK,IY))
                YMAX = MAX (YMAX,YNBLOCK(IBLOCK,IY))
93           CONTINUE
             DO 95 IX = 1, NXBLOCK(IBLOCK)
                XMIN = MIN (XMIN,XWBLOCK(IBLOCK,IX))
                XMAX = MAX (XMAX,XEBLOCK(IBLOCK,IX))
95           CONTINUE
90        CONTINUE

          CALL GRSSET (XMIN, XMAX, YMIN, YMAX)

96        CALL GR_CONTROL(ZRGNBN, INDGR, PLTITL,
     1       A1, A2, A3, A4, A5, A6, A7, A8, A9, A10)
C
C
```

```
C       PROCESS EACH BLOCK FOR NODES AND CELLS; SOME OF THE NODES
C       WILL BE DUPLICATE, BUT THE CELL WILL NOT BE
C
C       INITIALIZE THE NUMBER OF CELLS, NODES AND BOUNDARY CONDITION POINTERS
C
100     NBNDG2 = 0
        NNODG2 = 0
        NCELG2 = 0

        DO 110 IBLOCK = 1, NBLOCK
C         SET ALL THE GRID POINTS FOR THIS BLOCK
          CALL GNSEPB (IBLOCK)
          NNODG2 = NNODG2 + NNODEH
110     CONTINUE
C
C       CHECK FOR OVERFLOW IN NODE AND CELL ARRAYS
C
        IF(NNODG2 .GT. MNODG2) THEN
          ZER1 = NNODG2
          ZER2 = MNODG2
          CALL ERRORM (6,'GNBLOC','NNODG2',ZER1,'MNODG2',ZER2,JPRINT,
     1      'NUMBER OF NODES EXCEEDS ITS LIMIT')
        ENDIF
C
        IF(NCELG2 .GT. MCELG2) THEN
          ZER1 = NCELG2
          ZER2 = MCELG2
          CALL ERRORM (7,'GNBLOC','NCELG2',ZER1,'MCELG2',ZER2,JPRINT,
     1      'NUMBER OF CELLS EXCEEDS ITS LIMIT')
        ENDIF
C
C       REMOVE REDUNDANCY OF NODES AT THE BOUNDARIES, FIRST INITIALIZE
C       THE NODE KEEP ARRAY

        CALL GNREDN
C
C       INITIALIZE AUXILIARY CELL INFORMATION
C
        DO 120 ICELL = 1, NCELG2
            KAUXG2(ICELL) = 0
120     CONTINUE
C
C       SET UP THE MULTIPLE-GRID-LEVEL ARRAY FOR THE GLOBAL FINE LEVEL
C
        ILVLG2(1,0) = ILVLG2(2,-1) + 1
        ILVLG2(2,0) = NCELG2
        ILVLG2(3,0) = ILVLG2(2,0) - ILVLG2(1,0) + 1
C
C       INITIALIZE THE MULTIPLE-GRID-LEVEL ARRAY FOR ALL EMBEDDED MESHES
C
        DO 130 ILEVEL = 1, MLVLG2
          ILVLG2(1,ILEVEL) = NCELG2 + 1
          ILVLG2(2,ILEVEL) = NCELG2
          ILVLG2(3,ILEVEL) = 0
130     CONTINUE
C
C       SET UP THE BOUNDARY CONDITION POINTERS
```

```
C
      CALL GNBNDG
C
C     CORRECT THE BOUNDARY CONDITION POINTERS AT THE CORNERS
C     NOTE THAT MORE THAN FOUR CORNER POINTERS MAY CREATE A PROBLEM**
C
      DO 140 IB = 1, NBNDG2
        IEDGE = IBNDG2(4,IB)
        IF (IEDGE.EQ.2 .AND. IBCSW.GT.0) IBNDG2(5,IB) = IBCSW
        IF (IEDGE.EQ.4 .AND. IBCSE.GT.0) IBNDG2(5,IB) = IBCSE
        IF (IEDGE.EQ.6 .AND. IBCNE.GT.0) IBNDG2(5,IB) = IBCNE
        IF (IEDGE.EQ.8 .AND. IBCNW.GT.0) IBNDG2(5,IB) = IBCNW
140   CONTINUE
C
      GOTO 1300

1100  CONTINUE
C
C     READ THE DATA FROM A PREVIOUSLY DONE CASE WHICH WAS DONE BY
C     MANIPULATING THE BOUNDARIES

      OPEN (UNIT = JPNTRE, FILE = 'GNBLOC.DAT', STATUS = 'OLD')
      OPEN (UNIT = JGIVEN, FILE = 'GNBLOC.OUT', STATUS = 'NEW')
1110  FORMAT(11I7)

      READ (JPNTRE,1110) NNODG2, NCELG2, NBNDG2

      DO 1120 LC = 1, NCELG2
        READ (JPNTRE,1110) (ICELG2(IP,LC), IP=1,10), KAUXG2(LC)
1120  CONTINUE

      DO 1130 IB = 1, NBNDG2
        READ (JPNTRE,1110) (IBNDG2(IP,IB), IP = 1, 5)
1130  CONTINUE

      DO 1140 IN = 1, NNODG2
        READ (JPNTRE,1110) (NEIBG2(IP,IN), IP = 1, 4)
1140  CONTINUE

      DO 1150 LV = -MLVLG2, MLVLG2
        READ (JPNTRE,1110) (ILVLG2(IP,LV), IP = 1, 3)
1150  CONTINUE

      READ (JPNTRE,1110) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

      DO 1160 IN = 1, NNODG2
        READ(JPNTRE,1200) GEOMG2(1,IN),GEOMG2(2,IN)
1160  CONTINUE
1200  FORMAT(8E15.6)

1300  CONTINUE

      NCELA2 = NCELG2
      DO 1310 ICELL = 1, NCELA2
        ICELA2(ICELL) = ICELL
1310  CONTINUE
```

```
      GOTO 3100

2100  CONTINUE
C
C     READ THE DATA FROM A PREVIOUSLY WRITTEN CASE FROM JPNTRE.DAT

      OPEN (UNIT = JPNTRE, FILE = 'JPNTRE.DAT', STATUS = 'OLD',
     1      FORM = 'UNFORMATTED')

      CALL PSREDU
      PHI  = CHNGE2(1,1)
      RHOD = CHNGE2(1,2)
C
C     PLOT THE GRID
C
3100  CONTINUE

C     REMOVE THE CORNER BOUNDARY POINTERS FOR THE EMBEDDED DIAMONDS
C     OR WEDGES, SUCH AS IN "KUMAR" CASE

      IF (IFROMU .LT. 3) THEN
         CALL GNWEDG
         CALL GNPINJ
      ENDIF

      WRITE(JTERMO,80)
      READ (JTERMI,*) INDGR
      XMIN =  1000.
      XMAX = -1000.
      YMIN =  1000.
      YMAX = -1000.
      IF (INDGR .LT. 0) GOTO 6000
      DO 3120 INODE = 1, NNODG2
        ZX(INODE) = GEOMG2(1,INODE)
        ZY(INODE) = GEOMG2(2,INODE)
        XMIN      = MIN (XMIN,ZX(INODE))
        XMAX      = MAX (XMAX,ZX(INODE))
        YMIN      = MIN (YMIN,ZY(INODE))
        YMAX      = MAX (YMAX,ZY(INODE))
3120  CONTINUE

      IF (INDGR .NE. 0) GOTO 181
      INDGR = 22
      CALL GRSSET (XMIN, XMAX, YMIN, YMAX)
C
181   GRDUMY( 1) = NCELA2
      GRDUMY( 2) = NNODG2
      GRDUMY( 4) = NBNDG2
      GRDUMY( 5) = XMIN
      GRDUMY( 6) = XMAX
      GRDUMY( 7) = YMIN
      GRDUMY( 8) = YMAX
      GRDUMY(24) = 0.
      GRDUMY(25) = 0.
      ZZ7        = IFROMU

      CALL GNCONTR (ZRPLTG, INDGR, PLTITL,
```

```
     1     ICELG2,ICELA2,KAUXG2,ZX,ZY,GRDUMY,ZZ7,Z8,Z9,Z10)
C
6000    CONTINUE

        WRITE(JTERMO,6100)
6100    FORMAT(' INPUT ONE OF THE FOLLOWING'/
     1         5X, '1. WRITE OUTPUT FILE'/
     2         5X, '2. REPLOT DATA'/
     3         5X, '3. EXIT'/' ===> ',$)
        READ (JTERMI,*) IP
        IF (IP .EQ. 2) GOTO 3100

        IF (IP .NE. 1) STOP ' THE END'
        GOTO (7000,7000,8000), IFROMU
C
C       WRITE DOWN EVERYTHING
C
C       INTEGERS FROM G2COMN.INC

7000    WRITE (JGIVEN,1110) NNODG2, NCELG2, NBNDG2

        DO 310 LC = 1, NCELG2
          WRITE (JGIVEN,1110) (ICELG2(IP,LC), IP=1,10), KAUXG2(LC)
310     CONTINUE

        DO 320 IB = 1, NBNDG2
          WRITE (JGIVEN,1110) (IBNDG2(IP,IB), IP = 1, 5)
320     CONTINUE

        DO 330 IN = 1, NNODG2
          WRITE (JGIVEN,1110) (NEIBG2(IP,IN), IP = 1, 4)
330     CONTINUE

        DO 340 LV = -MLVLG2, MLVLG2
          WRITE (JGIVEN,1110) (ILVLG2(IP,LV), IP = 1, 3)
340     CONTINUE

        WRITE (JGIVEN,1110) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

        DO 350 IN = 1, NNODG2
          WRITE(JGIVEN,1200) GEOMG2(1,IN),GEOMG2(2,IN)
350     CONTINUE
        STOP ' THE END'

8000    APASKY(1) = PHI
        APASKY(2) = RHOD
        OPEN (UNIT = JGIVEN, FILE = 'JPNTWR.DAT', STATUS = 'NEW',
     1        FORM = 'UNFORMATTED')
        CALL PSWRTU(JGIVEN)
        STOP ' THE END'

        END
```

GNBNDG

```
      SUBROUTINE GNBNDG

      INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] HEXCOD.INC     '
      INCLUDE 'GNBLOC.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE SETS UP THE BOUNDARY CONDITION POINTERS FOR THE
C     INTEGRATED ASSEMBLY OF THE VARIOUS BLOCKS

C*********************************************************************
C
      DO 270 IBLOCK = 1, NBLOCK
C
C        CHECK THE SOUTHERN PHYSICAL SURFACE
C
         NBSURF = ISBLOCK(IBLOCK,1)
         IF (NBSURF .LT. 0) THEN
C
C           CHECK IF END POINTS ARE ALREADY IN THE ARRAY
C
            IBEG = 1
            IEND = NXBLOCK(IBLOCK)
            NBEG = IBS(IBLOCK,IBEG)
            NEND = IBS(IBLOCK,IEND)

            DO 190 IBNODE = 1, NBNDG2
              IF (IBNDG2(1,IBNODE) .EQ. NBEG) IBEG = IBEG + 1
              IF (IBNDG2(1,IBNODE) .EQ. NEND) IEND = IEND - 1
190         CONTINUE
C
C           CHECK IF THE FIRST LOCAL NODE IS SW CORNER
C
            IF (IBEG .EQ. 1) THEN
               NBCEL4 = NEIBG2(4,NBEG)
               IF (NBCEL4 .EQ. 0) THEN
                  NBNDG2                 = NBNDG2 + 1
                  IBNDG2(1,NBNDG2)       = NBEG
                  IBNDG2(2,NBNDG2)       = NEIBG2(3,NBEG)
                  IBNDG2(3,NBNDG2)       = 0
                  IBNDG2(4,NBNDG2)       = 2
                  IBNDG2(5,NBNDG2)       = ABS(NBSURF)
                  IBEG                   = IBEG + 1
                  PRESG2(1)              = NBNDG2
                  KAUXG2(IBNDG2(2,NBNDG2))=
     1                 IOR(KAUXG2(IBNDG2(2,NBNDG2)),KLOOOB)
               ENDIF
            ENDIF
C
C           CHECK IF THE LAST LOCAL NODE IS SE CORNER
C
```

```
            IF (IEND .EQ. NXBLOCK(IBLOCK)) THEN
               NBCEL3 = NEIBG2(3,NEND)
               IF (NBCEL3 .EQ. 0) THEN
                  NBNDG2                  = NBNDG2 + 1
                  IBNDG2(1,NBNDG2)        = NEND
                  IBNDG2(2,NBNDG2)        = NEIBG2(4,NEND)
                  IBNDG2(3,NBNDG2)        = 0
                  IBNDG2(4,NBNDG2)        = 4
                  IBNDG2(5,NBNDG2)        = ABS(NBSURF)
                  IEND                    = IEND - 1
                  PRESG2(2)               = NBNDG2
                  KAUXG2(IBNDG2(2,NBNDG2))=
     1                 IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0007)
               ENDIF
            ENDIF
C
C           NOW PROCESS ALL THE REST OF SOUTHERN NODES
C
            DO 200 IX = IBEG, IEND
              NBNDG2                  = NBNDG2 + 1
              IBNDG2(1,NBNDG2)        = IBS(IBLOCK,IX)
              IBNDG2(2,NBNDG2)        = NEIBG2(4,IBS(IBLOCK,IX))
              IBNDG2(3,NBNDG2)        = NEIBG2(3,IBS(IBLOCK,IX))
              IBNDG2(4,NBNDG2)        = 3
              IBNDG2(5,NBNDG2)        = ABS(NBSURF)
              KAUXG2(IBNDG2(2,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0003)
              KAUXG2(IBNDG2(3,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL0003)
200         CONTINUE

          ENDIF ! SOUTHERN SURFACE OF THIS BLOCK IS DONE
C
C         CHECK THE EASTERN PHYSICAL SURFACE
C
          NBSURF = ISBLOCK(IBLOCK,2)
          IF (NBSURF .LT. 0) THEN
C
C           CHECK IF END POINTS ARE ALREADY IN THE ARRAY
C
            IBEG = 1
            IEND = NYBLOCK(IBLOCK)
            NBEG = IBE(IBLOCK,IBEG)
            NEND = IBE(IBLOCK,IEND)

            DO 210 IBNODE = 1, NBNDG2
              IF (IBNDG2(1,IBNODE) .EQ. NBEG) IBEG = IBEG + 1
              IF (IBNDG2(1,IBNODE) .EQ. NEND) IEND = IEND - 1
210         CONTINUE
C
C           CHECK IF THE FIRST LOCAL NODE IS SE CORNER
C
            IF (IBEG .EQ. 1) THEN
               NBCEL1 = NEIBG2(1,NBEG)
               IF (NBCEL1 .EQ. 0) THEN
                  NBNDG2                  = NBNDG2 + 1
                  IBNDG2(1,NBNDG2)        = NBEG
```

```
                IBNDG2(2,NBNDG2)        = NEIBG2(4,NBEG)
                IBNDG2(3,NBNDG2)        = 0
                IBNDG2(4,NBNDG2)        = 4
                IBNDG2(5,NBNDG2)        = ABS(NBSURF)
                IBEG                    = IBEG + 1
                PRESG2(2)               = NBNDG2
                KAUXG2(IBNDG2(2,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0007)
             ENDIF
           ENDIF
C
C          CHECK IF THE LAST LOCAL NODE IS NE CORNER
C
           IF (IEND .EQ. NYBLOCK(IBLOCK)) THEN
             NBCEL4 = NEIBG2(4,NEND)
             IF (NBCEL4 .EQ. 0) THEN
                NBNDG2                  = NBNDG2 + 1
                IBNDG2(1,NBNDG2)        = NEND
                IBNDG2(2,NBNDG2)        = NEIBG2(1,NEND)
                IBNDG2(3,NBNDG2)        = 0
                IBNDG2(4,NBNDG2)        = 6
                IBNDG2(5,NBNDG2)        = ABS(NBSURF)
                IEND                    = IEND - 1
                PRESG2(3)               = NBNDG2
                KAUXG2(IBNDG2(2,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000E)
             ENDIF
           ENDIF
C
C          NOW PROCESS ALL THE REST OF EASTERN NODES
C
           DO 220 IY = IBEG, IEND
             NBNDG2                 = NBNDG2 + 1
             IBNDG2(1,NBNDG2)       = IBE(IBLOCK,IY)
             IBNDG2(2,NBNDG2)       = NEIBG2(1,IBE(IBLOCK,IY))
             IBNDG2(3,NBNDG2)       = NEIBG2(4,IBE(IBLOCK,IY))
             IBNDG2(4,NBNDG2)       = 5
             IBNDG2(5,NBNDG2)       = ABS(NBSURF)
             KAUXG2(IBNDG2(2,NBNDG2))=
     1              IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0006)
             KAUXG2(IBNDG2(3,NBNDG2))=
     1              IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL0006)
220        CONTINUE

         ENDIF ! EASTERN SURFACE OF THIS BLOCK IS DONE
C
C        CHECK THE NORTHERN PHYSICAL SURFACE
C
         NBSURF = ISBLOCK(IBLOCK,3)
         IF (NBSURF .LT. 0) THEN
C
C          CHECK IF END POINTS ARE ALREADY IN THE ARRAY
C
           IBEG = 1
           IEND = NXBLOCK(IBLOCK)
           NBEG = IBN(IBLOCK,IBEG)
           NEND = IBN(IBLOCK,IEND)
```

```
            DO 230 IBNODE = 1, NBNDG2
              IF (IBNDG2(1,IBNODE) .EQ. NBEG) IBEG = IBEG + 1
              IF (IBNDG2(1,IBNODE) .EQ. NEND) IEND = IEND - 1
230         CONTINUE
C
C           CHECK IF THE FIRST LOCAL NODE IS NW CORNER
C
            IF (IBEG .EQ. 1) THEN
               NBCEL1 = NEIBG2(1,NBEG)
               IF (NBCEL1 .EQ. 0) THEN
                  NBNDG2                  = NBNDG2 + 1
                  IBNDG2(1,NBNDG2)        = NBEG
                  IBNDG2(2,NBNDG2)        = NEIBG2(2,NBEG)
                  IBNDG2(3,NBNDG2)        = 0
                  IBNDG2(4,NBNDG2)        = 8
                  IBNDG2(5,NBNDG2)        = ABS(NBSURF)
                  IBEG                    = IBEG + 1
                  PRESG2(4)               = NBNDG2
                  KAUXG2(IBNDG2(2,NBNDG2))=
     1                 IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000D)
               ENDIF
            ENDIF
C
C           CHECK IF THE LAST LOCAL NODE IS NE CORNER
C
            IF (IEND .EQ. NXBLOCK(IBLOCK)) THEN
               NBCEL2 = NEIBG2(2,NEND)
               IF (NBCEL2 .EQ. 0) THEN
                  NBNDG2                  = NBNDG2 + 1
                  IBNDG2(1,NBNDG2)        = NEND
                  IBNDG2(2,NBNDG2)        = NEIBG2(1,NEND)
                  IBNDG2(3,NBNDG2)        = 0
                  IBNDG2(4,NBNDG2)        = 6
                  IBNDG2(5,NBNDG2)        = ABS(NBSURF)
                  IEND                    = IEND - 1
                  PRESG2(3)               = NBNDG2
                  KAUXG2(IBNDG2(2,NBNDG2))=
     1                 IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000E)
               ENDIF
            ENDIF
C
C           NOW PROCESS ALL THE REST OF NORTHERN NODES
C
            DO 240 IX = IBEG, IEND
              NBNDG2                  = NBNDG2 + 1
              IBNDG2(1,NBNDG2)        = IBN(IBLOCK,IX)
              IBNDG2(2,NBNDG2)        = NEIBG2(2,IBN(IBLOCK,IX))
              IBNDG2(3,NBNDG2)        = NEIBG2(1,IBN(IBLOCK,IX))
              IBNDG2(4,NBNDG2)        = 7
              IBNDG2(5,NBNDG2)        = ABS(NBSURF)
              KAUXG2(IBNDG2(2,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000C)
              KAUXG2(IBNDG2(3,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL000C)
240         CONTINUE
```

```
      ENDIF ! NORTHERN SURFACE OF THIS BLOCK IS DONE
C
C     CHECK THE WESTERN PHYSICAL SURFACE
C
      NBSURF = ISBLOCK(IBLOCK,4)
      IF (NBSURF .LT. 0) THEN
C
C        CHECK IF END POINTS ARE ALREADY IN THE ARRAY
C
         IBEG = 1
         IEND = NYBLOCK(IBLOCK)
         NBEG = IBW(IBLOCK,IBEG)
         NEND = IBW(IBLOCK,IEND)

         DO 250 IBNODE = 1, NBNDG2
           IF (IBNDG2(1,IBNODE) .EQ. NBEG) IBEG = IBEG + 1
           IF (IBNDG2(1,IBNODE) .EQ. NEND) IEND = IEND - 1
250      CONTINUE
C
C        CHECK IF THE FIRST LOCAL NODE IS SW CORNER
C
         IF (IBEG .EQ. 1) THEN
            NBCEL2 = NEIBG2(2,NBEG)
            IF (NBCEL2 .EQ. 0) THEN
               NBNDG2                  = NBNDG2 + 1
               IBNDG2(1,NBNDG2)        = NBEG
               IBNDG2(2,NBNDG2)        = NEIBG2(3,NBEG)
               IBNDG2(3,NBNDG2)        = 0
               IBNDG2(4,NBNDG2)        = 2
               IBNDG2(5,NBNDG2)        = ABS(NBSURF)
               IBEG                    = IBEG + 1
               PRESG2(1)               = NBNDG2
               KAUXG2(IBNDG2(2,NBNDG2))=
     1              IOR(KAUXG2(IBNDG2(2,NBNDG2)),KLOOOB)
            ENDIF
         ENDIF
C
C        CHECK IF THE LAST LOCAL NODE IS NW CORNER
C
         IF (IEND .EQ. NYBLOCK(IBLOCK)) THEN
            NBCEL3 = NEIBG2(3,NEND)
            IF (NBCEL3 .EQ. 0) THEN
               NBNDG2                  = NBNDG2 + 1
               IBNDG2(1,NBNDG2)        = NEND
               IBNDG2(2,NBNDG2)        = NEIBG2(2,NEND)
               IBNDG2(3,NBNDG2)        = 0
               IBNDG2(4,NBNDG2)        = 8
               IBNDG2(5,NBNDG2)        = ABS(NBSURF)
               IEND                    = IEND - 1
               PRESG2(4)               = NBNDG2
               KAUXG2(IBNDG2(2,NBNDG2))=
     1              IOR(KAUXG2(IBNDG2(2,NBNDG2)),KLOOOD)
            ENDIF
         ENDIF
C
C        NOW PROCESS ALL THE REST OF WESTERN NODES
C
```

```
            DO 260 IY = IBEG, IEND
              NBNDG2                  = NBNDG2 + 1
              IBNDG2(1,NBNDG2)        = IBW(IBLOCK,IY)
              IBNDG2(2,NBNDG2)        = NEIBG2(3,IBW(IBLOCK,IY))
              IBNDG2(3,NBNDG2)        = NEIBG2(2,IBW(IBLOCK,IY))
              IBNDG2(4,NBNDG2)        = 9
              IBNDG2(5,NBNDG2)        = ABS(NBSURF)
              KAUXG2(IBNDG2(2,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0009)
              KAUXG2(IBNDG2(3,NBNDG2))=
     1               IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL0009)
260         CONTINUE

          ENDIF ! WESTERN SURFACE OF THIS BLOCK IS DONE
270     CONTINUE
C
C       CHECK FOR OVERFLOW IN BOUNDARY NODE ARRAYS

        IF(NBNDG2 .GT. MBNDG2) THEN
          ZER1 = NBNDG2
          ZER2 = MBNDG2
          CALL ERRORM (8,'GNBLOC','NBNDG2',ZER1,'MBNDG2',ZER2,JPRINT,
     1      'NUMBER OF BOUNDARY NODES EXCEEDS ITS LIMIT')
        ENDIF
C
C       SET UP THE POINTERS FOR THE FOUR CORNER CELLS
C
        DO 290 IX = 1, 4
          ICOR  = NINT(PRESG2(IX))
          ICELL = IBNDG2(2,ICOR)
          IF (IX .EQ. 1) THEN
C           SW CORNER
            NODEBF = ICELG2(8,ICELL)
            NODEAF = ICELG2(4,ICELL)
          ELSE IF (IX .EQ. 2) THEN
C           SE CORNER
            NODEBF = ICELG2(2,ICELL)
            NODEAF = ICELG2(6,ICELL)
          ELSE IF (IX .EQ. 3) THEN
C           NE CORNER
            NODEBF = ICELG2(4,ICELL)
            NODEAF = ICELG2(8,ICELL)
          ELSE
C           NW CORNER
            NODEBF = ICELG2(6,ICELL)
            NODEAF = ICELG2(2,ICELL)
          ENDIF
          DO 280 IBOUND = 1, NBNDG2
            IF (IBNDG2(1,IBOUND) .EQ. NODEBF) NBCPG2(IX,1)=IBOUND
            IF (IBNDG2(1,IBOUND) .EQ. NODEAF) NBCPG2(IX,2)=IBOUND
280       CONTINUE
290     CONTINUE

        WRITE(6,*) ' NCELG2=',NCELG2
        WRITE(6,*) ' NNODG2=',NNODG2
        WRITE(6,*) ' NBNDG2=',NBNDG2
```

```
      RETURN
      END
```

GNCHAN

```
      SUBROUTINE GNCHAN (JCELL)

      INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
      INCLUDE 'GNBLOC.INC/LIST'
      DIMENSION MARKBN(MBNDG2)

C*********************************************************************
C
C     THIS SUBROUTINE CHANGES THE POINTERS FOR A SPECIFIED CELL
C
C*********************************************************************
C
      IF (JCELL .LE. 0 .OR. JCELL .GT. NCELG2) RETURN
      IP1 = ICELG2(2,JCELL)
      IP2 = ICELG2(4,JCELL)
      IP3 = ICELG2(6,JCELL)
      IP4 = ICELG2(8,JCELL)
      KX  = KAUXG2(JCELL)
      WRITE(6,1000)
      READ(5,*) IOPT
      IF (IOPT .GE. 5 .OR. IOPT .LE. 0) RETURN

      IF (IOPT .EQ. 1) THEN
         WRITE(6,1100) JCELL,KX
         READ(5,1200) KX
         KAUXG2(JCELL) = KX
         WRITE(6,1300) JCELL,KX
      ELSEIF (IOPT .EQ. 2) THEN
         WRITE(6,1400) JCELL,IP1,IP2,IP3,IP4,KX
         IF (KX .EQ. 0) RETURN
         KNODE1 = 0
         KNODE2 = 0
         KNODEC = 0
         DO 10 JBND = 1, NBNDG2
           IF (JCELL.EQ.IBNDG2(3,JBND)) KNODE2 = JBND
           IF (JCELL.EQ.IBNDG2(2,JBND) .AND. IBNDG2(3,JBND).NE.0)
     1                  KNODE1 = JBND
           IF (JCELL.EQ.IBNDG2(2,JBND) .AND. IBNDG2(3,JBND).EQ.0)
     1                  KNODEC = JBND
10       CONTINUE
         IF (KNODE1 .NE. 0)
     1      WRITE(6,1500) KNODE1, (IBNDG2(J,KNODE1),J=1,5)
         IF (KNODE2 .NE. 0)
     1      WRITE(6,1500) KNODE2, (IBNDG2(J,KNODE2),J=1,5)
         IF (KNODEC .NE. 0) THEN
            WRITE(6,1500) KNODEC, (IBNDG2(J,KNODEC),J=1,5)
            IF (IBNDG2(4,KNODEC) .EQ. 2) THEN
              WRITE(6,1600) NBCPG2(1,1), KNODEC, NBCPG2(1,2)
```

```
            ENDIF
          - IF (IBNDG2(4,KNODEC) .EQ. 4) THEN
              WRITE(6,1600) NBCPG2(2,1), KNODEC, NBCPG2(2,2)
            ENDIF
            IF (IBNDG2(4,KNODEC) .EQ. 6) THEN
              WRITE(6,1600) NBCPG2(3,1), KNODEC, NBCPG2(3,2)
            ENDIF
            IF (IBNDG2(4,KNODEC) .EQ. 8) THEN
              WRITE(6,1600) NBCPG2(4,1), KNODEC, NBCPG2(4,2)
            ENDIF
          ENDIF
          WRITE(6,1700)
          READ (5,*) KNCHAN
          IF (KNCHAN .LT. 0) THEN
            WRITE(6,*) '  INPUT POINTERS OF NEW BOUNDARY NODE'
            NBNDG2 = NBNDG2 + 1
            READ (5,*) (IBNDG2(J,NBNDG2),J=1,5)
            WRITE(6,1500) NBNDG2, (IBNDG2(J,NBNDG2),J=1,5)
          ENDIF
          IF (KNCHAN .EQ. KNODEC .OR. KNCHAN .EQ. KNODE1 .OR.
     1        KNCHAN .EQ. KNODE2) THEN
            IBNDG2(1,KNCHAN) = -9

C           DELETE ALL BOUNDARY CONDITION POINTERS MARKED FOR DELETE
C
            NNEW = 0
            DO 30 NOLD = 1, NBNDG2
              MARKBN(NOLD) = 0
              IF (IBNDG2(1,NOLD) .NE. -9) THEN
                NNEW = NNEW + 1
                MARKBN(NOLD) = NNEW
C               MOVE POINTER INFORMATION
                IF (NOLD .NE. NNEW) THEN
                  DO 20 J = 1, 5
                    IBNDG2(J,NNEW) = IBNDG2(J,NOLD)
20                CONTINUE
                ENDIF
              ENDIF
30          CONTINUE
C
C           RESET NUMBER OF BOUNDARY CONDITION POINTERS
C
            NBNDG2 = NNEW

            DO 50 IEDGE = 1, 4
              DO 40 IBND = 1, 2
                NBCPG2(IEDGE,IBND) = MARKBN(NBCPG2(IEDGE,IBND))
40            CONTINUE
50          CONTINUE
            WRITE(6,*) ' NCELG2=',NCELG2
            WRITE(6,*) ' NNODG2=',NNODG2
            WRITE(6,*) ' NBNDG2=',NBNDG2

          ENDIF
        ELSEIF (IOPT .EQ. 3) THEN
          WRITE(6,1400) JCELL,IP1,IP2,IP3,IP4,KX
          KNODE1 = 0
```

```
      KNODE2 = 0
      KNODEC = 0
      DO 60 JBND = 1, NBNDG2
        IF (JCELL.EQ.IBNDG2(3,JBND)) KNODE2 = JBND
        IF (JCELL.EQ.IBNDG2(2,JBND) .AND. IBNDG2(3,JBND).NE.0)
     1                  KNODE1 = JBND
        IF (JCELL.EQ.IBNDG2(2,JBND) .AND. IBNDG2(3,JBND).EQ.0)
     1                  KNODEC = JBND
60    CONTINUE
      IF (KNODE1 .NE. 0)
     1   WRITE(6,1500) KNODE1, (IBNDG2(J,KNODE1),J=1,5)
      IF (KNODE2 .NE. 0)
     1   WRITE(6,1500) KNODE2, (IBNDG2(J,KNODE2),J=1,5)
      IF (KNODEC .NE. 0) THEN
         WRITE(6,1500) KNODEC, (IBNDG2(J,KNODEC),J=1,5)
         IF (IBNDG2(4,KNODEC) .EQ. 2) THEN
           WRITE(6,1600) NBCPG2(1,1), KNODEC, NBCPG2(1,2)
         ENDIF
         IF (IBNDG2(4,KNODEC) .EQ. 4) THEN
           WRITE(6,1600) NBCPG2(2,1), KNODEC, NBCPG2(2,2)
         ENDIF
         IF (IBNDG2(4,KNODEC) .EQ. 6) THEN
           WRITE(6,1600) NBCPG2(3,1), KNODEC, NBCPG2(3,2)
         ENDIF
         IF (IBNDG2(4,KNODEC) .EQ. 8) THEN
           WRITE(6,1600) NBCPG2(4,1), KNODEC, NBCPG2(4,2)
         ENDIF
      ENDIF
      WRITE(6,1800)
      READ (5,*) KNCHAN
      WRITE(6,1500) KNCHAN, (IBNDG2(J,KNCHAN),J=1,5)
      WRITE(6,1900)
      READ (5,*) (IBNDG2(J,KNCHAN),J=1,5)
      WRITE(6,1500) KNCHAN, (IBNDG2(J,KNCHAN),J=1,5)
    ELSEIF (IOPT .EQ. 4) THEN
      WRITE(6,1400) JCELL,IP1,IP2,IP3,IP4,KX
      WRITE(6,1950)
      READ(5,*) IOPT1
      IF (IOPT1 .GE. 3 .OR. IOPT1 .LE. 0) RETURN

      IF (IOPT1 .EQ. 1) THEN
         WRITE(6,2000)
         READ (5,*) KNODE1
         WRITE(6,2100)
         READ (5,*) KNODE2
         WRITE(6,2200)
         DO 70 JQ = 1, NEQNFL
            WRITE(6,2300) JQ,DPENG2(JQ,KNODE1),DPENG2(JQ,KNODE2)
            DPENG2(JQ,KNODE1) = DPENG2(JQ,KNODE2)
70       CONTINUE
         JQ = 0
         WRITE(6,2300) JQ,PRESG2(KNODE1),PRESG2(KNODE2)
         WRITE(6,2300) JQ,TEMPG2(KNODE1),TEMPG2(KNODE2)
         PRESG2(KNODE1) = PRESG2(KNODE2)
         TEMPG2(KNODE1) = TEMPG2(KNODE2)
      ELSEIF (IOPT1 .EQ. 2) THEN
         WRITE(6,2400) IP1,DPENG2(4,IP1)
```

```
            WRITE(6,2400) IP2,DPENG2(4,IP2)
            WRITE(6,2400) IP3,DPENG2(4,IP3)
           - WRITE(6,2400) IP4,DPENG2(4,IP4)
            WRITE(6,*) ' INPUT THE ADDITIONAL AMOUNT OF ENERGY'
            READ (5,*) ADDENG
            DPENG2(4,IP1) = DPENG2(4,IP1) + ADDENG
            DPENG2(4,IP2) = DPENG2(4,IP2) + ADDENG
            DPENG2(4,IP3) = DPENG2(4,IP3) + ADDENG
            DPENG2(4,IP4) = DPENG2(4,IP4) + ADDENG
          ENDIF
        ENDIF
C
1000    FORMAT(1X,'INPUT ONE OF THE FOLLOWING'/
     1         5X,'1. CHANGE AUXILIARY POINTER'/
     2         5X,'2. DELETE/ADD BOUNARY POINTER'/
     3         5X,'3. CHANGE BOUNARY NODE POINTER'/
     4         5X,'4. CHANGE DEPENDENT VARIABLES AT A NODE'/
     5         5X,'9. EXIT'   / 5X,'    ===> ',$)
1100    FORMAT(5X,'AUX. POINTER OF CELL:',I5,5X,Z10/
     1        10X,'INPUT NEW AUX. POINTER'/' 12345678')
1200    FORMAT(Z8)
1300    FORMAT(5X,'AUX. POINTER OF CELL:',I5,5X,Z10)
1400    FORMAT(5X,'POINTERS OF CELL:',I5,5X,4I5,2X,Z10)
1500    FORMAT(7X,'BOUNDARY  NODE:',I5,5X,5I5)
1600    FORMAT(7X,'CORNER N. NODE:',5I5)
1700    FORMAT(5X,'INPUT THE BOUNDARY NODE TO BE DELETED'/
     1   5X,'INPUT < 0 IF A BOUNDARY NODE IS TO BE ADDED'/' ===> ',$)
1800    FORMAT(5X,'INPUT THE BOUNDARY NODE WHOSE POINTERS ARE TO BE'
     1           ' CHANGED'/' ===> ',$)
1900    FORMAT(5X,'INPUT THE NEW POINTERS FOR THIS BOUNDARY NODE',
     1           ' ===> ',$)
1950    FORMAT(1X,'INPUT ONE OF THE FOLLOWING'/5X,
     1       '1. ASSIGN DEPENDENT VARIABLES FROM ONE NODE TO OTHER'/5X,
     1       '2. ADD ENERGY (TEMPERATURE) TO THE CELL'/5X,
     5       '3. EXIT'/5X,'    ===> ',$)
2000    FORMAT(5X,'INPUT THE NODE WHOSE DEPENDENT VARIABLES ARE TO',
     1           ' BE CHANGED'/5X,' ===>',$)
2100    FORMAT(5X,'INPUT THE NODE WHOSE DEPENDENT VARIABLES ARE ',
     1           ' ALLOCATED TO THE PREVIOUS NODE'/5X,' ===>',$)
2200    FORMAT(/5X,'DPEN OLD',10X,'DPEN NEW')
2300    FORMAT(I5,2G15.6)
2400    FORMAT(I5,G15.6)

        RETURN
        END
```

GNCLP0

```
        SUBROUTINE GNCLP0 (ICELL)

        INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
```

```
      DIMENSION MEMBER(4)

C*********************************************************************
C
C       THIS SUBROUTINE COLLAPSES CELLS ADJACENT TO A SPECIFIED CELL
C       PROVIDED THAT IT FINDS A QUADRUPLE OF CELLS WHICH WERE PREVIOUSLY
C       CONSTRUCTED FROM A SINGLE CELL.
C
C*********************************************************************
C
        IF (ICELL .LE. 0 .OR. ICELL .GT. NCELG2) RETURN
        IDBGG2 = 3
        ISUPER = ICELG2(10,ICELL)
        IF (ISUPER .EQ. 0) RETURN
C
        IFIRST = ICELL - 5
        ILAST  = ICELL + 5
        NOELEM = 0

        DO 10 JCELL = IFIRST, ILAST
           IF ( ISUPER .EQ. ICELG2(10,JCELL) ) THEN
              NOELEM         = NOELEM + 1
              MEMBER(NOELEM) = JCELL
           ENDIF
           IF (NOELEM .EQ. 4) GOTO 20
10      CONTINUE
C
C       LESS THAN FOUR CELLS ARE FOUND
        RETURN
C
20      MEM1  = MEMBER(1)
        MEM2  = MEMBER(2)
        MEM3  = MEMBER(3)
        MEM4  = MEMBER(4)
        IWARN = 0
        CALL G2CLPO (MEM1, MEM2, MEM3, MEM4, ISUPER, IWARN)
        CALL G2NODE

        RETURN
        END
```

GNCONTR

```
        SUBROUTINE GNCONTR (GRPKG, INDGR1, PLTITL,
     1                      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
        INCLUDE '[PERVAIZ.GRAFIC1]GRCOMN.INC/LIST'
        INCLUDE '[PERVAIZ.GRAFIC1]MPCOMN.INC/LIST'
        INCLUDE 'GNBLOC.INC/LIST'
C
        INTEGER QSIZES(8)
        DIMENSION ICLINE(128), ICMARK(128), IMARKS(128), ICTEXT(128),
     3            ICFILL(20) , IBUNDLE(13), INGR(7)    ,
     2            XLINE(128) , YLINE(128) , XLINE1(128), YLINE1(128),
     1            XMARK(128) , YMARK(128) , RECTX(10)  , RECTY(10)  ,
```

```
     2           ALIMITS(6)
                 -
        CHARACTER GR_ORVILL_QMS*40   , INFO_STRING*80, JCHAR*1 ,
     1           QMS_DEFAULT_FILE*40, QSTRING(8)*64 , JCHAR2*1,
     2           METAFILE_DEFAULT*40, LNAME*40      , LNAME1*40,
     3           LNAME2*40          , PLTITL*(*)
C
        LOGICAL GR_TEST_BIT, IQ_REDRAW, SAVE_PORTRAIT
        SAVE GR_TEX_PORTRAIT
        DATA QMS_DEFAULT_FILE /'QMS.QMS'/
        DATA METAFILE_DEFAULT /'META.FIL'/
        DATA GR_TEX_PORTRAIT  /.TRUE./
C
C*********************************************************************
C
C     THIS SUBROUTINE CONTAINS THE CONTROL LOGIC FOR GRAPHICS WHEN
C     INTERACTIVE GRID GENERATION IS DESIRED
C
C       INDGR is a bit collection with the following meanings:
C       BIT     VALUE   OFF                     ON
C       1       1       Scales in common        Calculate scales
C       2       2       Independent scales      Dependent scales
C       3       4                               Draw axes
C       4       8                               Draw grid
C       5       16      Auto-hard copy          Interactive plotting
C       6       32      Put CFD logo on plots   No logo
C       7       64      Mouse menus             No mouse menus
C
C     GRPKG IS A USER SUPPLIED PLOTTING PACKAGE WHICH CREATES
C        AN IMAGE BY SUITABLE CALLS TO GR_DRAW, GR_MOVE, AND GR_ANNOTATE
C        THE CALLING SEQUENCE IS:
C            CALL GRPKG (IFUN,INDGR,PLTITL,ALIMITS,INFO_STRING,
C                        A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C       IFUN = 0         Initialize
C       IFUN = 1         Return XMIN,XMAX,YMIN,YMAX in ALIMITS(1)-ALIMITS(4)
C       IFUN = 2         Return INFO_STRING for X,Y in ALIMITS(1)-ALIMITS(2)
C                        for VALUE command.
C       IFUN = 3         Plot
C
C*********************************************************************
C
C     INITIALLY THERE IS NO ANNOTATION, LINES ETC.
C
        NANOT       = 0
        NLINES      = 0
        NMARKS      = 0
        NPOLYS      = 0
        INDGR       = INDGR1
        NCHOICE     = 22
        INDEXC_LINE = 1
        INDEXC_MARK = 1
        INDEXC_TEXT = 1
        INDEXC_FILL = 1
        Iaa         = ICHAR('a')
        Izz         = ICHAR('z')
        IOUTLN      = 0
        IFORMU      = NINT(A7)
```

```
C
C       Initialize user package
C
        CALL GRPKG (0,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
        DO I = 1,7
            INGR(I) = 0
            IF (GR_TEST_BIT(INDGR,I)) INGR(I) = 1
        ENDDO
C
C       SAVE INDICATOR FOR SCALING
C
        INDS=INGR(1)
C
C       CALCULATE MINIMUMS AND MAXIMUMS TO PLOT (FULL DATA)
C
30      CONTINUE
        CALL GRPKG (1,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
        XMIN = ALIMITS(1)
        XMAX = ALIMITS(2)
        YMIN = ALIMITS(3)
        YMAX = ALIMITS(4)

C
C       OPEN THE APPROPRIATE GRAPHICS MODE
C
50      CONTINUE
        IF (INGR(5).EQ.0) IDEVGR = 0
        CALL GR_MODE (1)
        CALL GR_MODE (0)
C
C       DRAW AXES AND SCALE
C
        CALL GR_AXES(PLTITL,XMIN,XMAX,YMIN,YMAX,INDS,INDGR)
C
C       PLOT WITH EXTERNALLY WRITTEN PLOTTER
C
        IF (IDEVGR.NE.41) CALL PLTON
        CALL GRPKG (3,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
        CALL GR_LINE_TYPE (0,0)
        IF (IDEVGR.NE.41) CALL PLTOFF
C
        DO 70 IANOT = 1, NANOT
           IF (IFLAGC .NE. 0) CALL GR_COLOR_INDEX(1,ICTEXT(IANOT))
           CALL GR_MOVE(XANOT(IANOT),YANOT(IANOT),0)
           CALL GR_ANNOTATE(CANOT(IANOT))
70      CONTINUE

        DO ILINE = 1,NLINES
           IF (IFLAGC .NE. 0) CALL GR_COLOR_INDEX(2,ICLINE(ILINE))
           CALL GR_MOVE(XLINE(ILINE),YLINE(ILINE),0)
           CALL GR_DRAW(XLINE1(ILINE),YLINE1(ILINE),0)
        ENDDO
```

```
      DO IMARK = 1,NMARKS
         IF (IFLAGC .NE. 0) CALL GR_COLOR_INDEX(3,ICMARK(IMARK))
         CALL GR_MOVE(XMARK(IMARK),YMARK(IMARK),IMARKS(IMARK))
      ENDDO

      DO IPOLY = 1,NPOLYS
         DO II = 1, NVERTP(IPOLY)
            RECTX(II) = XVERTP(IPOLY,II)
            RECTY(II) = YVERTP(IPOLY,II)
         ENDDO
         CALL GR_FILL( ICFILL(IPOLY), ISTYLE(IPOLY),
     1                 RECTX, RECTY , NVERTP(IPOLY) )
      ENDDO
C
C     MENU FOR INTERACTIVE GRAPHICS
C
100   CONTINUE
C
      IF (IDEVGR.EQ.41) CALL GKS$UPDATE_WS (1,0)
      IF (INGR(5).EQ.0) THEN
        READ(JINGR,'(A)') JCHAR
      ELSE IF(IDEVGR.EQ.41.AND.INGR(7).EQ.0) THEN
        CALL GNGKIN(1,NCHOICE,JCHAR,XX,YY)
C       CONVERT BACK TO VIRTUAL COORDINATES
        XCURSG = XMINGR+(XX-XOFFSET)*(XMAXGR-XMINGR)/GRXTICKS
        YCURSG = YMINGR+(YY-YOFFSET)*(YMAXGR-YMINGR)/GRYTICKS
      ELSE
        CALL GR_CURSOR('A B C D L M O P Q R S T V W X ?',
     &       JCHAR,XCURSG,YCURSG)
C       Convert to upper case
        IJ = ICHAR(JCHAR)
        IF (IJ.GE.Iaa .AND. IJ.LE.Izz) JCHAR = CHAR(IJ - 32)
      ENDIF
C
      IF(JCHAR.EQ.'A') GOTO 200
      IF(JCHAR.EQ.'B') GOTO 500
      IF(JCHAR.EQ.'C') WRITE (JOUTGR,'(A/)') (' ',I=1,33)
      IF(JCHAR.EQ.'D') GOTO 1400
      IF(JCHAR.EQ.'G') GOTO 300
C     IF(JCHAR.EQ.'E') GOTO 1900
      IF(JCHAR.EQ.'H') GOTO 1800
      IF(JCHAR.EQ.'I') GOTO 1900
      IF(JCHAR.EQ.'L') GOTO 600
      IF(JCHAR.EQ.'M') GOTO 700
      IF(JCHAR.EQ.'N') GOTO 1700
      IF(JCHAR.EQ.'O') GOTO 800
      IF(JCHAR.EQ.'P') GOTO 1500
      IF(JCHAR.EQ.'Q') GOTO 900
      IF(JCHAR.EQ.'R') GOTO 400
      IF(JCHAR.EQ.'S') THEN
          CALL GR_REAL('Enter new symbol size',ASIZE)
          CALL GR_SET_SYMBOL_SIZE (ASIZE)
      ENDIF
      IF(JCHAR.EQ.'T') GOTO 1200
      IF(JCHAR.EQ.'U') GOTO 1600
      IF(JCHAR.EQ.'V') GOTO 1000
      IF(JCHAR.EQ.'W') GOTO 1100
```

```
      IF(JCHAR.EQ.'o') GOTO 1300
      IF(JCHAR.EQ.'v') GOTO 2100
      IF(JCHAR.EQ.'X') THEN
C
C     CLOSE THE GRAPHICS BEFORE EXITING and remove annotations
C
          CALL GR_MODE(-1)
          NANOT = 0
          RETURN
      ENDIF
C
      IF(JCHAR.EQ.'?') THEN
          WRITE (JOUTGR,*) '     A = Add feature'
          WRITE (JOUTGR,*) '     B = Blowup'
          WRITE (JOUTGR,*) '     C = Clear text plane'
          WRITE (JOUTGR,*) '     D = Detailed grid information'
          WRITE (JOUTGR,*) '     H = Eliminate holes -- G2DIVO'
          WRITE (JOUTGR,*) '     I = Eliminate islands -- G2CLPO'
          WRITE (JOUTGR,*) '     L = Lasergrafix mode'
          WRITE (JOUTGR,*) '     M = Min/Max'
          WRITE (JOUTGR,*) '     N - Locate a Node or cell'
          WRITE (JOUTGR,*) '     O = Original scales'
          WRITE (JOUTGR,*) '     P = Put thick outline for blocks'
          WRITE (JOUTGR,*) '     Q = Query'
          WRITE (JOUTGR,*) '     R = Remove feature'
          WRITE (JOUTGR,*) '     S = Set new symbol size'
          WRITE (JOUTGR,*) '     T = TeX Output'
          WRITE (JOUTGR,*) '     U = Undo (change) auxilliary pointers'
          WRITE (6,*) ' S/U - Subdivide/Undivide elements. Hit X to do it.'
          WRITE (JOUTGR,*) '     V = Value'
          WRITE (JOUTGR,*) '     W = Window'
          WRITE (JOUTGR,*) '     X = eXit'
          WRITE (JOUTGR,*) '     ? = Help'
      ENDIF
      GOTO 100
C
C          -----------
C*****A*****ADD FEATURE
C          -----------
C
200   CONTINUE

      IF (INGR(5).EQ.0) THEN
         READ(JINGR,'(A)') JCHAR2
      ELSE IF(IDEVGR.EQ.41.AND.INGR(7).EQ.0) THEN
         CALL GKIN2 (1,JCHAR2,'ADD')
      ELSE
         CALL GR_CURSOR('A L M P ?                    ',
     &                  JCHAR2,XCURSG,YCURSG         )
C        Convert to upper case
         IJ = ICHAR(JCHAR2)
         IF (IJ.GE.Iaa .AND. IJ.LE.Izz) JCHAR2 = CHAR(IJ - 32)
      ENDIF

      IF (IFLAGC .NE. 0) THEN
        CALL GKS$INQ_INDIV_ATTB (IERRST, LINE_TYPE, WIDTH_LINE,
     1             INDEXC_LINE, MARK_TYPE, SIZE_MARK, INDEXC_MARK,
```

```
     2              IFONT, IPRECISION, EXPFAC, SPACING, INDEXC_TEXT,
     3     -        INTSTYLE, INDEX_FILL, INDEXC_FILL, IBUNDLE)
        ENDIF
C
        IF(JCHAR2.EQ.'A') GOTO 220
        IF(JCHAR2.EQ.'L') GOTO 240
        IF(JCHAR2.EQ.'M') GOTO 250
        IF(JCHAR2.EQ.'P') GOTO 260
        IF(JCHAR2.EQ.'?') THEN
          WRITE (JOUTGR,210)
        ENDIF
210     FORMAT ('     A = Annotate'/
     1          '     L = Draw Line'/
     2          '     M = Draw Marker (Symbol)'/
     3          '     P = Draw Polygon (Fill Area)'/
     4          '     ? = Help'/)
        GOTO 100

C**********A**** Annotate
220     IF(NANOT.EQ.128) THEN
            WRITE(JOUTGR,230)
            GOTO 100
        ENDIF
230     FORMAT (' ONLY 128 FEATURES ALLOWED ')
C
        IF (IDEVGR.EQ.0) THEN
            CALL GR_REAL ('Enter X position',XCURSG)
            CALL GR_REAL ('Enter Y position',YCURSG)
        ENDIF
        NANOT=NANOT+1
        XANOT(NANOT)=XCURSG
        YANOT(NANOT)=YCURSG
        ICTEXT(NANOT)=INDEXC_TEXT + 1
        CALL GR_ASCII(' Enter annotation',64,CANOT(NANOT))
C
        CALL GR_MOVE(XANOT(NANOT),YANOT(NANOT),0)
        CALL GR_ANNOTATE(CANOT(NANOT))
C
        GOTO 100
C
C**********D**** Draw a line
C
C       Get other end of line
C
240     IF (IDEVGR.EQ.0) GOTO 100
        IF (NLINES.EQ.128) THEN
            WRITE (JOUTGR,*) ' Only 128 lines allowed'
            GOTO 100
        ENDIF
        NLINES = NLINES + 1
        XLINE(NLINES) = XCURSG
        YLINE(NLINES) = YCURSG
        ICLINE(NLINES)=INDEXC_LINE + 1
        IF(IDEVGR.EQ.41) THEN
            WRITE(JOUTGR,*) ' INPUT OTHER END'
C
C           INPUT 2ND CORNER OF LINE USING GKLOC (LINE TYPE LOCATOR)
```

```
            XXC=XOFFSET+GRXTICKS*(XCURSG-XMINGR)/(XMAXGR-XMINGR)
            YYC=YOFFSET+GRYTICKS*(YCURSG-YMINGR)/(YMAXGR-YMINGR)
            CALL GKLOC(1,1,4,XXC,YYC,XXNEW,YYNEW)
C           WRITE(6,*) 'XXC,YYC',XXC,YYC
C           WRITE(6,*) 'XXNEW,YYNEW',XXNEW,YYNEW
C
C           CONVERT BACK TO VIRTUAL COORDINATES
            XNEW=XMINGR+(XXNEW-XOFFSET)*(XMAXGR-XMINGR)/GRXTICKS
            YNEW=YMINGR+(YYNEW-YOFFSET)*(YMAXGR-YMINGR)/GRYTICKS
        ELSE
            CALL GR_CURSOR('OTHER END                          ',
     1                        JCHAR,XNEW,YNEW)
        ENDIF
C
        XLINE1(NLINES) = XNEW
        YLINE1(NLINES) = YNEW
        CALL GR_MOVE(XCURSG,YCURSG,0)
        CALL GR_DRAW(XNEW,YNEW,0)
        GOTO 100
C**********M**** Draw a Marker (Symbol)
250     CONTINUE
        IF (IDEVGR.EQ.0) GOTO 100
        IF (NMARKS.EQ.128) THEN
            WRITE (JOUTGR,*) ' Only 128 markers allowed'
            GOTO 100
        ENDIF
        CALL GR_INTEGER('Enter symbol number',ISYMBOL)
        NMARKS = NMARKS + 1
        XMARK(NMARKS) = XCURSG
        YMARK(NMARKS) = YCURSG
        ICMARK(NMARKS)=INDEXC_MARK + 1
        IMARKS(NMARKS)=ISYMBOL
C
        CALL GR_MOVE(XCURSG,YCURSG,ISYMBOL)
        GOTO 100
c
C**********P**** Draw a Polygon (Fill area)
260     CONTINUE
        if (idevgr .ne. 41) goto 100
        IF (NPOLYS.EQ.20) THEN
            WRITE (JOUTGR,*) ' Only 20 polygons allowed'
            GOTO 100
        ENDIF
        NPOLYS = NPOLYS + 1
        ICFILL(NPOLYS)=INDEXC_FILL + 1
        CALL GR_COLOR_INT (1,7,NPOLYS,XX,YY)
        GOTO 100
C
C           -----------------
C*****G*****Color index setup
C           -----------------
C
300     CONTINUE
        if (idevgr .ne. 41) goto 100
        IF(INGR(7).EQ.0) THEN
           CALL GKIN2 (1,JCHAR2,'INDEX')
        ELSE
```

```
          CALL GR_CURSOR('A L M P ?                          ',
     &                   JCHAR2,XCURSG,YCURSG          )
C         Convert to upper case
          IJ = ICHAR(JCHAR2)
          IF (IJ.GE.Iaa .AND. IJ.LE.Izz) JCHAR2 = CHAR(IJ - 32)
        ENDIF

        IF(JCHAR2.EQ.'A') CALL GR_COLOR_INT (1,3,IDUM,XX,YY)
        IF(JCHAR2.EQ.'L') CALL GR_COLOR_INT (1,4,IDUM,XX,YY)
        IF(JCHAR2.EQ.'M') CALL GR_COLOR_INT (1,5,IDUM,XX,YY)
        IF(JCHAR2.EQ.'P') CALL GR_COLOR_INT (1,6,IDUM,XX,YY)
        IF(JCHAR2.EQ.'?') THEN
          WRITE (JOUTGR,210)
        ENDIF
        GOTO 100
C
C           --------------
C*****R*****Remove Feature
C           --------------
C
400     CONTINUE

        IF (INGR(5).EQ.0) THEN
           READ(JINGR,'(A)') JCHAR2
        ELSE IF(IDEVGR.EQ.41.AND.INGR(7).EQ.0) THEN
           CALL GKIN2 (1,JCHAR2,'REMOVE')
        ELSE
           CALL GR_CURSOR('A L M P ?                          ',
     &                    JCHAR2,XCURSG,YCURSG          )
C          Convert to upper case
           IJ = ICHAR(JCHAR2)
           IF (IJ.GE.Iaa .AND. IJ.LE.Izz) JCHAR2 = CHAR(IJ - 32)
        ENDIF
C
        IF(JCHAR2.EQ.'A') then
          NANOT = MAX (0,NANOT-1)
          GOTO 50
        ENDIF
        IF(JCHAR2.EQ.'L') THEN
          NLINES = MAX (0,NLINES-1)
          GOTO 50
        ENDIF
        IF(JCHAR2.EQ.'M') THEN
          NMARKS = MAX (0,NMARKS-1)
          GOTO 50
        ENDIF
        IF(JCHAR2.EQ.'P') THEN
          NPOLYS = MAX (0,NPOLYS-1)
          GOTO 50
        ENDIF
        IF(JCHAR2.EQ.'?') THEN
          WRITE (JOUTGR,210)
        ENDIF
        GOTO 100
c
C           ------
C*****B*****BLOWUP
```

```
C          ------
C
500     CALL GR_REAL('ENTER XMIN',XMIN)
        CALL GR_REAL('ENTER XMAX',XMAX)
        CALL GR_REAL('ENTER YMIN',YMIN)
        CALL GR_REAL('ENTER YMAX',YMAX)
C
        IF(ABS(XMIN-XMAX).LT.1.0E-25) GOTO 100
        IF(ABS(YMIN-YMAX).LT.1.0E-25) GOTO 100
C
C       RE-SCALE AND RE-PLOT BASED ON ABOVE VALUES (DO NOT NORMALIZE
C       SCALES)
C
        CALL GR_SET_SCALE(XMIN,XMAX,YMIN,YMAX)
        INDS=0
        GOTO 50
C
C          ------------------
C*****L*****Lasergraphics mode
C          ------------------
C
C       SET TO HARD COPY MODE
C
600     CONTINUE
        WRITE (JOUTGR,610) QMS_DEFAULT_FILE
610     FORMAT ('  Default file is ',A)
        CALL GR_ASCII(' Enter file name (QUIT to quit)',40,LNAME)
        IF (LNAME.EQ.' ') LNAME=QMS_DEFAULT_FILE
        IF (LNAME.EQ.'QUIT'.OR.LNAME.EQ.'quit') GOTO 100
        JHRDGR = 47
C
        OPEN(UNIT=JHRDGR,FILE=LNAME,STATUS='NEW',ERR=620)
        GOTO 640
C
620     WRITE(JOUTGR,630)
630     FORMAT(' Error opening file, try a different name.')
        GOTO 600
640     CONTINUE
C
        CALL GR_MODE(2)
C
C       DRAW THE IMAGE TO THE RASTER AND CLOSE IT
C
        CALL GR_AXES(PLTITL,XMIN,XMAX,YMIN,YMAX,INDS,INDGR)
        CALL GRPKG (3,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
c       IF (IOUTLN .EQ. 1) THEN
          CALL GR_LINE_TYPE (5,0)
          DO 641 IBL = 1, NBLOCK
            CALL GR_MOVE (XSBLOCK(IBL,1),YSBLOCK(IBL,1),0)
            CALL GR_DRAW (XEBLOCK(IBL,1),YEBLOCK(IBL,1),0)
            CALL GR_DRAW (XEBLOCK(IBL,NYBLOCK(IBL)),
     1                   YEBLOCK(IBL,NYBLOCK(IBL)),0)
            CALL GR_DRAW (XNBLOCK(IBL,1),YNBLOCK(IBL,1),0)
            CALL GR_DRAW (XSBLOCK(IBL,1),YSBLOCK(IBL,1),0)
641       CONTINUE
```

```
c       ENDIF
        CALL GR_LINE_TYPE (0,0)
C           -
C       Just in case, call PLTOFF
C
        CALL PLTOFF
        DO 650 IANOT=1,NANOT
            CALL GR_MOVE(XANOT(IANOT),YANOT(IANOT),0)
            CALL GR_ANNOTATE(CANOT(IANOT))
650     CONTINUE
        DO ILINE = 1,NLINES
            CALL GR_MOVE(XLINE(ILINE),YLINE(ILINE),0)
            CALL GR_DRAW(XLINE1(ILINE),YLINE1(ILINE),0)
        ENDDO
        DO IMARK = 1,NMARKS
           CALL GR_MOVE(XMARK(IMARK),YMARK(IMARK),IMARKS(IMARK))
        ENDDO
C
        CALL GR_MODE (-2)
        JHRDGR = 0
C
C       RESET TO TERMINAL MODE
C
        CALL GR_MODE (1)
        WRITE(JOUTGR,660) LNAME
660     FORMAT(' Hard copy sent to ',A40)
        GOTO 100
C
C           -------
C*****M*****MIN/MAX
C           -------
C
C       CALCULATE MINIMUMS AND MAXIMUMS TO PLOT (FULL DATA)
C
700     CONTINUE
        ALIMITS(1) = XMINGR
        ALIMITS(2) = XMAXGR
        ALIMITS(3) = YMINGR
        ALIMITS(4) = YMAXGR
        ALIMITS(5) = 0.
        ALIMITS(6) = 0.
        INFO_STRING = ' '

        CALL GRPKG (1,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
        WRITE(JOUTGR,710) (ALIMITS(I),I=1,4)
710     FORMAT(' MINIMUM AND MAXIMUM DATA VALUES:'/
     &      10X,'XMIN=',G15.7,10X,'XMAX=',G15.7 /
     &      10X,'YMIN=',G15.7,10X,'YMAX=',G15.7 /)
C
        IF (INFO_STRING .NE. ' ') THEN
           WRITE (JOUTGR,1020) INFO_STRING
           WRITE(JOUTGR,720) ALIMITS(5), ALIMITS(6)
        ENDIF
720     FORMAT(  10X,'ZMIN=',G15.7,10X,'ZMAX=',G15.7 )
C
```

```
      GOTO 100
C
C          ---------------
C*****O*****ORIGINAL SCALES
C          ---------------
C
800   INDS = 1
      GOTO 30
C
C          -----
C*****Q*****QUERY
C          -----
C
900   CONTINUE
      IOLDINGR1 = INGR(1)
      IQ_REDRAW = .FALSE.
C
C%%%%%%%%%%%%%%%%%%%%%%%%%FOR VSII WORKSTATION TERMINAL%%%%%%%%%%%%%%%%%%%
C     SET UP QUERY MENU
910   WRITE(QSTRING(1),920) INGR(1)
      WRITE(QSTRING(2),925) INGR(2)
      WRITE(QSTRING(3),930) INGR(3)
      WRITE(QSTRING(4),935) INGR(4)
      WRITE(QSTRING(5),940) GRXTICKS
      WRITE(QSTRING(6),945) GRYTICKS
920   FORMAT(' 1. CALC SCALES   ',I5)
925   FORMAT(' 2. OPEN SCALES   ',I5)
930   FORMAT(' 3. DRAW AXES     ',I5)
935   FORMAT(' 4. DRAW GRID     ',I5)
940   FORMAT(' 5. X AXIS TICKS  ',I5)
945   FORMAT(' 6. Y AXIS TICKS  ',I5)
      QSTRING(7) = ' 7. QMS OPTIONS'
      QSTRING(8) = ' 8. EXIT FROM QUERY'
      QSIZES(1)=23
      QSIZES(2)=23
      QSIZES(3)=23
      QSIZES(4)=23
      QSIZES(5)=23
      QSIZES(6)=23
      QSIZES(7)=23
      QSIZES(8)=23
C
C     REQUEST QUERY INPUT
C
      IF(IDEVGR.EQ.41.AND.INGR(7).EQ.0) THEN
          CALL GKCHOIC(1,8,QSTRING,QSIZES,IOPT)
      ELSE
          WRITE(JOUTGR,'(A)') (QSTRING(I),I=1,8)
          CALL GR_INTEGER('ENTER OPTION NUMBER',IOPT)
      ENDIF
C
C
      IF(IOPT.GE.1 .AND.IOPT.LE.4) INGR(IOPT) = 1 - INGR(IOPT)
      IF(IOPT.EQ.5) THEN
          CALL GR_INTEGER ('Enter X Axis ticks (1-20)',ITX)
          CALL GR_SET_TICKS (ITX,GRYTICKS)
          IQ_REDRAW = .TRUE.
```

```
        ENDIF
        IF(IQPT.EQ.6) THEN
            CALL GR_INTEGER ('Enter Y Axis ticks (1-20)',ITY)
            CALL GR_SET_TICKS (GRXTICKS,ITY)
            IQ_REDRAW = .TRUE.
        ENDIF
C
C       EXIT OUT OF QUERY
C
        IF (IOPT.EQ.8.OR.IOPT.EQ.0) THEN
            INLOW = MOD(INDGR,16)
            INNEW = INGR(1) + 2*INGR(2) + 4*INGR(3) + 8*INGR(4)
            INDGR = 16*(INDGR/16)
            INDGR = INDGR + INNEW
            IF (IOLDINGR1.NE.INGR(1)) THEN
                IQ_REDRAW = .TRUE.
                INDS = INGR(1)
            ENDIF
            IF (IQ_REDRAW .OR. INNEW.NE.INLOW) GOTO 50
            GOTO 100
C
C
        ELSE IF (IOPT.EQ.7) THEN
C
C       Do the QMS options
C
950         WRITE(QSTRING(1),955) 1 - INGR(6)
            QSTRING(2) = ' 2. Landscape Mode'
            IF (GR_PORTRAIT) QSTRING(2) = ' 2. Portrait Mode'
            WRITE (QSTRING(3),960) GR_QMS_SCALE_FACTOR
            QSTRING(4) = ' 4. Change default QMS file'
            QSTRING(5) = ' 5. Exit QMS options'
            QSIZES(1)=27
            QSIZES(2)=27
            QSIZES(3)=27
            QSIZES(4)=27
            QSIZES(5)=27
C
C       REQUEST INPUT
C
            IF(IDEVGR.EQ.41.AND.INGR(7).EQ.0) THEN
                CALL GKCHOIC(1,5,QSTRING,QSIZES,IOPT)
            ELSE
                WRITE(JOUTGR,'(A)') (QSTRING(I),I=1,5)
                CALL GR_INTEGER('ENTER OPTION NUMBER',IOPT)
            ENDIF
            IF (IOPT.EQ.1) INGR(6) = 1 - INGR(6)
            IF (IOPT.EQ.2) GR_PORTRAIT = .NOT. GR_PORTRAIT
            IF (IOPT.EQ.3) THEN
                CALL GR_REAL ('Enter new QMS scale factor',FF)
                CALL GR_SET_QMS_SCALE (FF)
            ENDIF
            IF (IOPT.EQ.4) THEN
                WRITE (JOUTGR,610) QMS_DEFAULT_FILE
                CALL GR_ASCII(' Enter new default name',40,
     &              QMS_DEFAULT_FILE)
                IF (QMS_DEFAULT_FILE.EQ.' ') QMS_DEFAULT_FILE = 'QMS.QMS'
```

```
            ENDIF
            IF (IOPT.EQ.5.OR.IOPT.EQ.0) THEN
            -   IF (INGR(6).EQ.1) CALL GR_SET_BIT(INDGR,6)
                IF (INGR(6).EQ.0) CALL GR_CLEAR_BIT(INDGR,6)
                GOTO 910
            ENDIF
            GOTO 950
          ENDIF
C
C     ELSE RETURN FOR ANOTHER QUERY
C
955     FORMAT(' 1. Draw CFD Logo       ',I1)
960     FORMAT(' 3. Scale Factor   ',F5.3)
        GOTO 910
C
C           -----
C*****V*****VALUE
C           -----
C
1000    IF (IDEVGR.EQ.0) GOTO 100
        CALL GR_MOVE (XCURSG,YCURSG,-2)
C
        INFO_STRING = ' '
        WRITE(JOUTGR,1010) XCURSG,YCURSG
1010    FORMAT(' CURSOR LOCATION: X ',G14.7/
     &          '                 Y ',G14.7)
        ALIMITS (1) = XCURSG
        ALIMITS (2) = YCURSG
        CALL GRPKG (2,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
        IF (INFO_STRING .NE. ' ') WRITE (JOUTGR,1020) INFO_STRING
1020    FORMAT (A)
C
        GOTO 100
C
C           ------
C*****W*****WINDOW
C           ------
C
C       GET OTHER CORNER OF WINDOW
C
1100    IF (IDEVGR.EQ.0) GOTO 500
        IF(IDEVGR.EQ.41) THEN
            WRITE(JOUTGR,*) ' INPUT OPPOSITE CORNER'
C
C           INPUT 2ND CORNER OF WINDOW USING GKLOC (BOX TYPE LOCATOR)
            XXC=XOFFSET+GRXTICKS*(XCURSG-XMINGR)/(XMAXGR-XMINGR)
            YYC=YOFFSET+GRYTICKS*(YCURSG-YMINGR)/(YMAXGR-YMINGR)
            CALL GKLOC(1,1,5,XXC,YYC,XXNEW,YYNEW)
C
C           CONVERT BACK TO VIRTUAL COORDINATES
            XNEW=XMINGR+(XXNEW-XOFFSET)*(XMAXGR-XMINGR)/GRXTICKS
            YNEW=YMINGR+(YYNEW-YOFFSET)*(YMAXGR-YMINGR)/GRYTICKS
        ELSE
            CALL GR_CURSOR ('OPPOSITE CORNER                    ',
     1                      JCHAR,XNEW,YNEW)
        ENDIF
```

```
C
1110    XMIN = MIN(XNEW,XCURSG)
        XMAX = MAX(XNEW,XCURSG)
        YMIN = MIN(YNEW,YCURSG)
        YMAX = MAX(YNEW,YCURSG)
C
        IF(ABS(XMIN-XMAX).LT.1.0E-25) GOTO 100
        IF(ABS(YMIN-YMAX).LT.1.0E-25) GOTO 100
C
C       If dependent scaling is enabled then keep it.
C
        IF (INGR(2).EQ.1) THEN
            DELTAX = XMAX - XMIN
            DELTAY = YMAX - YMIN
            TIC = MAX (DELTAX/GRXTICKS, DELTAY/GRYTICKS)
            XMAX = XMIN + GRXTICKS*TIC
            YMAX = YMIN + GRYTICKS*TIC
        ENDIF
C
C       RESCALE AND REPLOT WITH ABOVE WINDOW
C
        CALL GR_SET_SCALE(XMIN,XMAX,YMIN,YMAX)
C
        INDS=0
        GOTO 50
C
C           ------------------
C*****T*****TeX quality output
C           ------------------
C
1200    CONTINUE
C
C       Get scale factors and modes
C
        JHRDGR = 47
        CALL GR_ASCII (' Enter file name (no type)',40,LNAME)
        IF (LNAME.EQ.' ') LNAME = 'GRAFIC_TEX'
        CALL GR_REAL ('Enter plot scale factor',TEX_SCALE)
        SAVE_SCALE = GR_QMS_SCALE_FACTOR
        CALL GR_SET_QMS_SCALE(TEX_SCALE)
        SAVE_PORTRAIT = GR_PORTRAIT
        GR_PORTRAIT = .TRUE.
C
C       Open the files and set the mode
C
        ILEN = INDEX(LNAME,' ')
        LNAME1 = LNAME(1:ILEN-1) // '.TEX'
        LNAME2 = LNAME(1:ILEN-1) // '.QMS'
        OPEN(UNIT=JHRDGR,FILE=LNAME2,STATUS='NEW',ERR=1210)
        OPEN(UNIT=JHRDGR+1,FILE=LNAME1,STATUS='NEW',ERR=1210)
        GOTO 1220
1210    WRITE (JOUTGR,*) ' Error opening files.'
        GOTO 1280
1220    CONTINUE
C
        CALL GR_MODE(3)
C
```

```
C       Initialize the file
C
        WRITE (JHRDGR+1,1230) GR_SCALE_FACTOR
1230    FORMAT(' {\setlength{\unitlength}{',F5.3,'in}')
        WRITE (JHRDGR+1,1240) FLOAT(GRXTICKS)+2.5,FLOAT(GRYTICKS)+2
1240    FORMAT(' \scriptsize \begin{picture}(',F6.3,',',F6.3,
     &      ')(-1,-1)')
C
C       DRAW THE IMAGE
C
        CALL GR_AXES(PLTITL,XMIN,XMAX,YMIN,YMAX,INDS,INDGR)
        CALL GRPKG (3,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
        CALL GR_LINE_TYPE (0,0)
C
C       Just in case, call PLTOFF
C
        CALL PLTOFF
        DO IANOT=1,NANOT
            CALL GR_MOVE(XANOT(IANOT),YANOT(IANOT),0)
            CALL GR_ANNOTATE(CANOT(IANOT))
        ENDDO

        DO ILINE = 1,NLINES
            CALL GR_MOVE(XLINE(ILINE),YLINE(ILINE),0)
            CALL GR_DRAW(XLINE1(ILINE),YLINE1(ILINE),0)
        ENDDO

        DO IMARK = 1,NMARKS
           CALL GR_MOVE(XMARK(IMARK),YMARK(IMARK),IMARKS(IMARK))
        ENDDO
C
C       Write the include information and close the file
C
        WRITE (JHRDGR+1,1250) GRYTICKS,GRXTICKS,GRYTICKS
1250    FORMAT (' \put(0,',I2,') {\begin{picture}(',I2,',',I2,')')
        WRITE (JHRDGR+1,1260) LNAME2(1:ILEN+3)
1260    FORMAT (' \special{include(',A,
     &      ' origin noorigin noorient nofree fortran norelative)}'/
     &      '\end{picture}}',/,'\end{picture}}')
        CLOSE (JHRDGR+1)
        WRITE(JOUTGR,1270) LNAME1(1:ILEN+3)
1270    FORMAT(/' TeX file ',A,' created.'/)
C
C       Error and normal return
C
1280    CONTINUE
        CALL GR_MODE (-3)
        JHRDGR = 0
        CALL GR_MODE (1)
        CALL GR_SET_QMS_SCALE(SAVE_SCALE)
        GR_PORTRAIT = SAVE_PORTRAIT
        GOTO 100
C
C           ---------------
C*****o*****METAFILE OUTPUT
C           ---------------
```

```
1300    CONTINUE
        IF (IDEVGR.NE.41) GOTO 100
        WRITE (JOUTGR,610) METAFILE_DEFAULT
        CALL GR_ASCII(' Enter file name (QUIT to quit)',40,LNAME)
        IF (LNAME.EQ.' ') LNAME = METAFILE_DEFAULT
        IF (LNAME.EQ.'QUIT'.OR.LNAME.EQ.'quit') GOTO 100
C       METFIL = 97
C       OPEN(UNIT=METFIL,FILE=LNAME,STATUS='NEW',ERR=1320)
C       GOTO 1340
C1320   WRITE(JOUTGR,630)
C       GOTO 1300
C
1340    CALL GKS$OPEN_WS(2,LNAME,2)
        CALL GKS$ACTIVATE_WS(2)
c       CALL GR_MODE (0)
        CALL GR_AXES(PLTITL,XMIN,XMAX,YMIN,YMAX,INDS,INDGR)
        CALL GRPKG (3,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
        DO IANOT = 1, NANOT
           IF (IFLAGC .NE. 0) CALL GR_COLOR_INDEX(1,ICTEXT(IANOT))
           CALL GR_MOVE(XANOT(IANOT),YANOT(IANOT),0)
           CALL GR_ANNOTATE(CANOT(IANOT))
        ENDDO
C
        DO ILINE = 1,NLINES
           IF (IFLAGC .NE. 0) CALL GR_COLOR_INDEX(2,ICLINE(ILINE))
           CALL GR_MOVE(XLINE(ILINE),YLINE(ILINE),0)
           CALL GR_DRAW(XLINE1(ILINE),YLINE1(ILINE),0)
        ENDDO
C
        DO IMARK = 1,NMARKS
           IF (IFLAGC .NE. 0) CALL GR_COLOR_INDEX(3,ICMARK(IMARK))
           CALL GR_MOVE(XMARK(IMARK),YMARK(IMARK),IMARKS(IMARK))
        ENDDO
C
        DO IPOLY = 1,NPOLYS
           IF (IFLAGC .NE. 0) CALL GR_COLOR_INDEX(4,ICFILL(IPOLY))
           DO II = 1, NVERTP(IPOLY)
              RECTX(II) = XVERTP(IPOLY,II)
              RECTY(II) = YVERTP(IPOLY,II)
           ENDDO
           CALL GKS$SET_FILL_INT_STYLE(ISTYLE(IPOLY))
           CALL GKS$FILL_AREA(NVERTP(IPOLY),RECTX,RECTY)
        ENDDO
C
        CALL GKS$DEACTIVATE_WS (2)
        CALL GKS$CLOSE_WS (2)
        GOTO 100
C
C           --------------------------------
C*****D*****DETAILED INFORMATION ABOUT GRIDS
C           --------------------------------
C
1400    IF (IDEVGR.EQ.0) GOTO 100
        ALIMITS(6) = -1.
        CALL GR_MOVE (XCURSG,YCURSG,-2)
```

```
C
        INFO_STRING = ' '
        ALIMITS (1) = XCURSG
        ALIMITS (2) = YCURSG
        CALL GRPKG (2,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
        IF (INFO_STRING .NE. ' ') WRITE (JOUTGR,1020) INFO_STRING
        JCELL = NINT(ALIMITS(5))
        CALL GNDEBG (JCELL)
        GOTO 100
C
C           ---------------------------
C*****P*****PUT THICK OUTLINE OF BLOCKS
C           ---------------------------
C
1500    IF (IDEVGR .EQ. 0) GOTO 100
        IF (IFROMU .NE. 1) GOTO 100
        IOUTLN      = 1
        CALL GR_LINE_TYPE (5,0)
        DO 1510 IBL = 1, NBLOCK
           CALL GR_MOVE (XSBLOCK(IBL,1),YSBLOCK(IBL,1),0)
           CALL GR_DRAW (XEBLOCK(IBL,1),YEBLOCK(IBL,1),0)
           CALL GR_DRAW (XEBLOCK(IBL,NYBLOCK(IBL)),
     1                   YEBLOCK(IBL,NYBLOCK(IBL)),0)
           CALL GR_DRAW (XNBLOCK(IBL,1),YNBLOCK(IBL,1),0)
           CALL GR_DRAW (XSBLOCK(IBL,1),YSBLOCK(IBL,1),0)
1510    CONTINUE
        CALL GR_LINE_TYPE (0,0)

        GOTO 100
C
C           ---------------------------
C*****U*****UNDO (CHANGE) AUX. POINTERS
C           ---------------------------
C
1600    IF (IDEVGR.EQ.0) GOTO 100
        ALIMITS(6) = -1.
        CALL GR_MOVE (XCURSG,YCURSG,0)
C
        INFO_STRING = ' '
        ALIMITS (1) = XCURSG
        ALIMITS (2) = YCURSG
        CALL GRPKG (2,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &      A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
        JCELL = NINT(ALIMITS(5))
        CALL GNCHAN(JCELL)
        GOTO 100
C
C           -----------
C*****N*****LOCATE NODE
C           -----------
C
1700    IF (IDEVGR.EQ.0) GOTO 100
        WRITE (JOUTGR,1710)
1710    FORMAT(
```

```
     1     5X,'INPUT NODE (NEGATIVE) OR CELL (POSITIVE) TO BE LOCATED'/
     2     5X,'ALSO INPUT SYMBOL TO BE PLOTTED')
           READ(5,*) JNODE,IISYM
           IISYM = -ABS(IISYM)
           CALL GNLNOD (JNODE, IISYM)
           GOTO 100
C
C          -------------------------
C*****H*****REMOVE HOLES; GRID DIVIDE
C          -------------------------
C
1800       IF (IDEVGR.EQ.0) GOTO 100
           ALIMITS(6) = -1.
           CALL GR_MOVE (XCURSG,YCURSG,0)
C
           INFO_STRING = ' '
           ALIMITS (1) = XCURSG
           ALIMITS (2) = YCURSG
           CALL GRPKG (2,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &          A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
           JCELL = NINT(ALIMITS(5))
           CALL G2DIVO(JCELL,0)
           CALL A2CEWC
           GOTO 100
C
C          ----------------------------
C*****I*****REMOVE ISLANDS; GRID COLLAPSE
C          ----------------------------
C
1900       IF (IDEVGR.EQ.0) GOTO 100
           ALIMITS(6) = -1.
           CALL GR_MOVE (XCURSG,YCURSG,0)
C
           INFO_STRING = ' '
           ALIMITS (1) = XCURSG
           ALIMITS (2) = YCURSG
           CALL GRPKG (2,INDGR,PLTITL,ALIMITS,INFO_STRING,
     &          A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
C
           JCELL = NINT(ALIMITS(5))
           CALL GNCLPO(JCELL)
           CALL A2CEWC
           GOTO 100
C
C          -------------------
C*****V*****SPECIAL DUMMY FILES
C          -------------------
C
2100       CONTINUE
           CALL GNDUMY
           GOTO 100

C          RETURN
           END
C          to be added later
C*****N*****MOVE NODE
```

```
C
C       IF(JCHAR.EQ.'N') GOTO 2000
C2000   CONTINUE
C       WRITE(6,2310) XCURSG,YCURSG
C       DMIN = 1.E30
C       NMIN = 0
C       DO 2010 I = 1,NN
C           DD = (X(1,I)-XCURSG)**2 + (X(2,I)-YCURSG)**2
C           IF (DD.LT.DMIN) THEN
C               NMIN = I
C               DMIN = DD
C           ENDIF
C2010   CONTINUE
C       CALL MY_CURSOR1(' NEW POSITION',JCHAR,XNEW,YNEW)
C       X(1,NMIN) = XNEW
C       X(2,NMIN) = YNEW
C       IF (JCHAR.EQ.'N'.OR.JCHAR.EQ.'n') THEN
C           CALL GR_CURSOR(' NEXT NODE     ',JCHAR,XCURSG,YCURSG)
C           GOTO 2000
C       ENDIF
C       GOTO 100
C
C*****S/U***** SUBDIVIDE AN AREA / COARSEN AN AREA
C
C       IF(JCHAR.EQ.'S') GOTO 2200
C       IF(JCHAR.EQ.'U') GOTO 2200
C2200   CONTINUE
C       IF (JCHAR.EQ.'U') THEN
C           ICFL = 1
C       ELSE
C           ICFL = 0
C       ENDIF
C       NP = 1
C       CALL GRMOVE (XCURSG,YCURSG,0)
C       XD(1,1) = XCURSG
C       XD(2,1) = YCURSG
C       DO WHILE (JCHAR.NE.'X' .AND. JCHAR.NE.'x' .AND. NP.LT.20)
C           NP = NP + 1
C           CALL MY_CURSOR1 ('NEXT CORNER     ',JCHAR,XD(1,NP),XD(2,NP))
C           CALL GR_DRAW (XD(1,NP),XD(2,NP),0)
C       ENDDO
C       IF (NP.LT.3) GOTO 100
C       IF (ICFL.EQ.0) THEN
C           CALL GDEMBED (XD,IEMBED,NP,IEMBTYPE)
C       ELSE
C           CALL GDCOARSE (XD,IEMBED,NP)
C       ENDIF
C       GOTO 50
```

GNDEBG

```
      SUBROUTINE GNDEBG (JCELL)

      INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
```

```
      INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
      INCLUDE 'GNBLOC.INC/LIST'
      CHARACTER*8 TYPINF

C*********************************************************************
C
C     THIS SUBROUTINE WRITES DETAILED INFORMATION FOR A SPECIFIED CELL
C
C*********************************************************************
C
      IF (JCELL .LE. 0 .OR. JCELL .GT. NCELG2) RETURN
      IP1 = ICELG2(2,JCELL)
      IP2 = ICELG2(4,JCELL)
      IP3 = ICELG2(6,JCELL)
      IP4 = ICELG2(8,JCELL)
      KX  = KAUXG2(JCELL)
      WRITE(6,10) JCELL,IP1,IP2,IP3,IP4,KX
10    FORMAT(5X,'POINTERS OF CELL:',I5,5X,4I5,2X,Z10)
      WRITE(6,20)
20    FORMAT(5X,'INPUT ONE OF THE FOLLOWING'/
     1       5X,'  1. POINTER INFORMATION'/
     2       5X,'  2. DEPENDENT VARIABLE INFORMATION'/
     3       5X,'  ==> ',$)

      READ (5,*) INFORM

      IF (INFORM .EQ. 1) THEN
         WRITE(6,40) IP1,(NEIBG2(J,IP1),J=1,4)
         WRITE(6,40) IP2,(NEIBG2(J,IP2),J=1,4)
         WRITE(6,40) IP3,(NEIBG2(J,IP3),J=1,4)
         WRITE(6,40) IP4,(NEIBG2(J,IP4),J=1,4)
         KNODEC = 0
         DO 30 JBND = 1, NBNDG2
            IF (IP1 .EQ. IBNDG2(1,JBND))
     1             WRITE(6,50) JBND, (IBNDG2(J,JBND),J=1,5)
            IF (IP2 .EQ. IBNDG2(1,JBND))
     1             WRITE(6,50) JBND, (IBNDG2(J,JBND),J=1,5)
            IF (IP3 .EQ. IBNDG2(1,JBND))
     1             WRITE(6,50) JBND, (IBNDG2(J,JBND),J=1,5)
            IF (IP4 .EQ. IBNDG2(1,JBND))
     1             WRITE(6,50) JBND, (IBNDG2(J,JBND),J=1,5)
            IF (JCELL.EQ.IBNDG2(2,JBND) .AND. IBNDG2(3,JBND).EQ.0)
     1                 KNODEC = JBND
30       CONTINUE
         IF (KNODEC .NE. 0) THEN
            IF (IBNDG2(4,KNODEC) .EQ. 2) THEN
              WRITE(6,60) NBCPG2(1,1), KNODEC, NBCPG2(1,2)
            ENDIF
            IF (IBNDG2(4,KNODEC) .EQ. 4) THEN
              WRITE(6,60) NBCPG2(2,1), KNODEC, NBCPG2(2,2)
            ENDIF
            IF (IBNDG2(4,KNODEC) .EQ. 6) THEN
              WRITE(6,60) NBCPG2(3,1), KNODEC, NBCPG2(3,2)
            ENDIF
            IF (IBNDG2(4,KNODEC) .EQ. 8) THEN
              WRITE(6,60) NBCPG2(4,1), KNODEC, NBCPG2(4,2)
            ENDIF
```

```
         ENDIF
C          -
       ELSE IF (INFORM .EQ. 2) THEN
          TYPINF = '  X-NODE'
          WRITE(6,70) TYPINF, GEOMG2(1,IP1), GEOMG2(1,IP2),
     1                        GEOMG2(1,IP3), GEOMG2(1,IP4)
          TYPINF = '  Y-NODE'
          WRITE(6,70) TYPINF, GEOMG2(2,IP1), GEOMG2(2,IP2),
     1                        GEOMG2(2,IP3), GEOMG2(2,IP4)
          MAXEQ  = MIN (10, NEQNFL)
          TYPINF = ' DPEN SW'
          WRITE(6,80) TYPINF, PRESG2(IP1), TEMPG2(IP1),
     1                (DPENG2(IS,IP1),IS=1,MAXEQ)
          TYPINF = ' DPEN SE'
          WRITE(6,80) TYPINF, PRESG2(IP2), TEMPG2(IP2),
     1                (DPENG2(IS,IP2),IS=1,MAXEQ)
          TYPINF = ' DPEN NE'
          WRITE(6,80) TYPINF, PRESG2(IP3), TEMPG2(IP3),
     1                (DPENG2(IS,IP3),IS=1,MAXEQ)
          TYPINF = ' DPEN NW'
          WRITE(6,80) TYPINF, PRESG2(IP4), TEMPG2(IP4),
     1                (DPENG2(IS,IP4),IS=1,MAXEQ)
       ENDIF

40     FORMAT(7X,'NEIGHBOURS OF:',I5,5X,4I5)
50     FORMAT (7X,'BOUNDARY  NODE:',I5,5X,5I5)
60     FORMAT (7X,'CORNER N. NODE:',5I5)
70     FORMAT(2X,A8,2X,4G15.5)
80     FORMAT(2X,A8,2X,4G15.5/12X,4G15.5/12X,4G15.5)

       RETURN
       END
```

GNDUMY

```
       SUBROUTINE GNDUMY

       INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
       INCLUDE 'GNBLOC.INC/LIST'

C**********************************************************************
C
C      THIS SUBROUTINE IS A DUMMY ROUTINE; ANY OTHER ROUTINE CAN BE
C      SUBSTITUTED FOR IT IF SOME SPECIAL PROCESSING IS TO BE DONE
C
C**********************************************************************
C
       WRITE(6,*) ' INPUT THE MINIMUM TEMPERATURE'
       READ (5,*) TEMPMN
       DO 10 INODE = 1, NNODG2
          IF (TEMPG2(INODE) .LE. TEMPMN) THEN
```

```
            XNODE = GEOMG2(1,INODE)
            YNODE = GEOMG2(2,INODE)
            CALL GR_MOVE(XNODE,YNODE,-2)
         ENDIF
10       CONTINUE
         WRITE(6,*) ' IF YOU WHICH TO CHANGE VALUES INPUT 1'
         READ (5,*) ICHAN

         IF (ICHAN .EQ. 1) THEN
            WRITE(6,*) ' INPUT THE NODE WHOSE VALUES ARE TO USED'
            READ (5,*) JNODE
            DO 30 INODE = 1, NNODG2
               IF (TEMPG2(INODE) .LE. TEMPMN) THEN
                  XNODE = GEOMG2(1,INODE)
                  YNODE = GEOMG2(2,INODE)
                  IF (XNODE.LT.1.2 .AND. XNODE.GT.0.55) THEN
                    IF (YNODE.LT.0.2 .AND. YNODE.GT.-0.2) THEN
                      DO 20 IEQ = 1, NEQNFL
                         DPENG2(IEQ,INODE) = DPENG2(IEQ,JNODE)
20                    CONTINUE
                      PRESG2(INODE) = PRESG2(JNODE)
                      TEMPG2(INODE) = TEMPG2(JNODE)
                    ENDIF
                  ENDIF
                  CALL GR_MOVE(XNODE,YNODE,-2)
               ENDIF
30          CONTINUE
         ENDIF


         RETURN
         END
```

GNGKIN

```
         SUBROUTINE GNGKIN (WSID,NCHOICE,CHAR,XX,YY)

         parameter (mchoic=25)
         INCLUDE '[PERVAIZ.GRAFIC1]GRCOMN.INC'
         INTEGER WSID,SIZES(mchoic),CHOICE
         CHARACTER*30 STRING(mchoic)
         CHARACTER*1 CHAR
         CHARACTER*25 CHOICE_STRING

         DATA CHOICE_STRING /'ABDHILMNOPQRSTUVWGFEX*KYZ'/
C        DATA CHOICE_STRING /'1234567890123456789012345'/
C
C*************************************************************************
C
C        THIS SUBROUTINE WRITES A MENU TO THE VSII SCREEN, ASKS FOR
C        CHOICE INPUT, AND IF NECESSARY ASKS FOR LOCATOR INPUT.
C
C                PARAMETER:
```

```
C       WSID    - LOGICAL WORKSTATION IDENTIFIER
C       CHAR    - CHOICE LETTER (OUTPUT)
C       XX,YY   - LOCATOR POINT (OUTPUT)
C
C*****************************************************************
C
C       MAKE MENU
        STRING(1)='ADD FEATURE'
        SIZES (1)=11
        STRING(2)='BLOWUP'
        SIZES (2)=6
        STRING(3)='DETAILED INFO'
        SIZES (3)=16
        STRING(4)='GRID DIVIDE'
        SIZES (4)=16
        STRING(5)='GRID COLLAPSE'
        SIZES (5)=16
        STRING(6)='LASERGRAFIX MODE'
        SIZES (6)=16
        STRING(7)='MIN/MAX'
        SIZES (7)=7
        STRING(8)='LOCATE NODE OR CELL'
        SIZES (8)=19
        STRING(9)='ORIGINAL SCALES'
        SIZES (9)=15
        STRING(10)='THICK OUTLINE OF BLOCKS'
        SIZES (10)=23
        STRING(11)='QUERY'
        SIZES (11)=5
        STRING(12)='REMOVE FEATURE'
        SIZES (12)=17
        STRING(13)='SET SYMBOL SIZE'
        SIZES (13)=15
        STRING(14)='TeX OUTPUT'
        SIZES (14)=10
        STRING(15)='CHANGE AUX. POINTER'
        SIZES (15)=19
        STRING(16)='VALUE'
        SIZES (16)=5
        STRING(17)='WINDOW'
        SIZES (17)=6
        STRING(18)='SET FEATURE COLOR INDEX'
        SIZES (18)=23
C
C       SET THE NUMBER OF BASIC CHOICES (FOR GR_CONTROL)
C
        NBASE  = 18

        STRING(19)='SET COLOR INDEX IN LUT'
        SIZES (19)=22
        STRING(20)='GET COLOR REP FROM LUT'
        SIZES (20)=22
        STRING(21)='EXIT'
        SIZES (21)=4
        STRING(22)='SPECIAL'
        SIZES (22)=7
C
```

```
C
C       INQUIRE MENU CHOICE
        CALL GKCHOIC (WSID,NCHOICE,STRING,SIZES,CHOICE)
C
        CHAR = CHOICE_STRING(CHOICE:CHOICE)
        XX = 0.
        YY = 0.
C
C
        IF (CHAR .EQ. 'X') RETURN

C       FIND CURSOR LOCATION FOR CHOICE = (A,D,U,V,W)

        IF(INDEX('ADHIUVW',CHAR).NE.0) THEN
           WRITE(JOUTGR,*) ' Input initial cursor location'
           CALL GKLOC (WSID,1,2,0.5,0.5,XX,YY)
C          WRITE(6,*) XX,YY
        ENDIF
C
C       SPECIAL PROCESSING

        IF(INDEX('*',CHAR).NE.0) THEN
           CALL GRSPEC (WSID,CHAR)
           RETURN
        ENDIF

C       DO INTERACTIVE COLOR MANIPULATIONS
        IF (CHOICE .GT. NBASE) THEN
           IGOTO = CHOICE - NBASE
           CALL GR_COLOR_INT (WSID,IGOTO,IDUM,XX,YY)
        ENDIF

        RETURN
        END
```

GNLNOD

```
        SUBROUTINE GNLNOD (JNODE, IISYM)
C
        INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
        INCLUDE '[PERVAIZ.GRAFIC1]GRCOMN.INC/LIST'
        INCLUDE '[PERVAIZ.GRAFIC1]MPCOMN.INC/LIST'
        INCLUDE 'GNBLOC.INC/LIST'
        DIMENSION RECTX(4), RECTY(4)

        IF (JNODE .LT. 0) THEN
C          THE GIVEN POINT IS A NODE
           INODE = -JNODE
           IF (INODE .EQ. 0 .OR. INODE .GT. NNODG2) RETURN
           XCURSG = GEOMG2(1,INODE)
           YCURSG = GEOMG2(2,INODE)
           CALL GR_MOVE (XCURSG,YCURSG,IISYM)
        ELSE
```

```
C          THE GIVEN POINT IS A CELL
           ICELL = JNODE
           IF (ICELL .EQ. 0 .OR. ICELL .GT. NCELG2) RETURN
           KSW    = ICELG2(2,ICELL)
           KSE    = ICELG2(4,ICELL)
           KNE    = ICELG2(6,ICELL)
           KNW    = ICELG2(8,ICELL)
           XSW    = GEOMG2(1,KSW)
           XSE    = GEOMG2(1,KSE)
           XNE    = GEOMG2(1,KNE)
           XNW    = GEOMG2(1,KNW)
           YSW    = GEOMG2(2,KSW)
           YSE    = GEOMG2(2,KSE)
           YNE    = GEOMG2(2,KNE)
           YNW    = GEOMG2(2,KNW)
           XCURSG = 0.25*(XSW + XSE + XNE + XNW)
           YCURSG = 0.25*(YSW + YSE + YNE + YNW)
           IF (IFLAGC .NE. 0) THEN
              RECTX(1) = XSW
              RECTX(2) = XSE
              RECTX(3) = XNE
              RECTX(4) = XNW
              RECTY(1) = YSW
              RECTY(2) = YSE
              RECTY(3) = YNE
              RECTY(4) = YNW
              ISYM     = ABS(IISYM)
              CALL GR_FILL (ISYM,1,RECTX,RECTY,4)
           ELSE
              CALL GR_MOVE (XCURSG,YCURSG,IISYM)
           ENDIF
        ENDIF
C
C       RETURN
        END
```

GNPINJ

```
        SUBROUTINE GNPINJ

        INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] HEXCOD.INC      '
        DIMENSION MARKBN(MBNDG2)
        CHARACTER*1 YESNO

C***********************************************************************
C
C       THIS SUBROUTINE IS USEFUL TO REMOVE THE EXTRA CORNER BOUNDARY
C       NODES WHEN THERE ARE EMBEDDED PARALLEL INJECTORS IN THE FLOW-
C       FIELD.  THESE CORNER POINTS ARE REGARDED AS THE DIRECHLET POINTS,
C       FOR EXAMPLE, IN THE "INJECTOR" CASE.
C
C***********************************************************************
```

```
C
C       INITIALIZE THE NUMBER OF INTERIOR POINTS
C
        KOUNTI = 0
C
C       LOCATE ALL THE INTERIOR CORNER NODES.  THESE NODES ARE DEFINED
C       AS THE WESTERN INLET BOUNDARIES IN THE INITIAL GRIDS

        DO 10 JBND = 1, NBNDG2

           INODE  = IBNDG2(1,JBND)
           IEDGE  = IBNDG2(4,JBND)
           IBCTYP = IBNDG2(5,JBND)

C          CHECK WESTERN BOUNDARY; THIS WOULD BE A REAL WESTERN BOUNDARY
C          IF THERE ARE TWO OR LESS NON-ZERO NEIGHBOUR CELLS OF THE NODE
C          "INODE", ELSE IT WOULD A REGULAR BOUNDRY NODE.  KOUNTN IS THE
C          COUNTER FOR NON-ZERO NEIGHBOUR CELLS

           KOUNTN = 0
           IF (IEDGE .EQ. 9 .AND. IBCTYP .EQ. 2) THEN
              NBSW = NEIBG2(1,INODE)
              NBSE = NEIBG2(2,INODE)
              NBNE = NEIBG2(3,INODE)
              NBNW = NEIBG2(4,INODE)
              IF (NBSW .NE. 0) KOUNTN = KOUNTN + 1
              IF (NBSE .NE. 0) KOUNTN = KOUNTN + 1
              IF (NBNE .NE. 0) KOUNTN = KOUNTN + 1
              IF (NBNW .NE. 0) KOUNTN = KOUNTN + 1

              IF (KOUNTN .EQ. 3) THEN
                 KOUNTI = KOUNTI + 1
                 MARKBN(KOUNTI) = JBND
              ENDIF
           ENDIF

10      CONTINUE

C       WRITE DOWN ALL THE INTERIOR CORNER BOUNDARY NODES AND QUERY IF
C       ANY OF THEM HAVE TO BE CHANGED

        IF (KOUNTI .EQ. 0) RETURN
        WRITE(6,20)
20      FORMAT(/5X,'THE FOLLOWING INTERIOR CORNER BOUNDARY NODES',
     1             ' ARE FOUND FOR PARELLEL INJECTION'/)

        DO 30 KOUNT = 1, KOUNTI
           JBND = MARKBN(KOUNT)
           WRITE(6,40) JBND, (IBNDG2(J,JBND),J=1,5)
30      CONTINUE
40      FORMAT(7X,'BOUNDARY  NODE:',I5,5X,5I5)

        WRITE(6,50)
50      FORMAT(/5X,'WANT TO CHANGE POINTERS OF ANY OF THE CORNER',
     1             ' BOUNDARY NODES')

        READ (5,60) YESNO
```

```
60      FORMAT(A)

        IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y') GOTO 70
        RETURN
C
C       INITIALIZE THE NUMBER OF INTERIOR BOUNDARY NODES TO BE CHANGED
C
70      KOUNTD = 0
C
C       SEE IF ANY OF THE CORNER BOUNDARY NODES ARE TO BE DELETED
C
        DO 90 KOUNT = 1, KOUNTI
           JBND = MARKBN(KOUNT)
           WRITE(6,100)
           WRITE(6,40) JBND, (IBNDG2(J,JBND),J=1,5)
           READ (5,60) YESNO
           IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y') THEN
               KOUNTD = KOUNTD + 1
               INODE = IBNDG2(1,JBND)
               IONE  = IBNDG2(2,JBND)
               ITWO  = IBNDG2(3,JBND)
               IF (ITWO .EQ. 0) GOTO 90
               I1SW = ICELG2(2,IONE)
               I1NW = ICELG2(8,IONE)
               I2SW = ICELG2(2,ITWO)
               I2NW = ICELG2(8,ITWO)
C              LOCATE THE CELL (B/W IONE AND ITWO) WITH NO BOUNDARY NODES
C              THIS CELL WILL BE THE FIRST CELL POINTER, THE OTHER CELL
C              POINTER OF THE BOUNDARY NODE WILL BECOME ZERO
               K1SW = 0
               K1NW = 0
               K2SW = 0
               K2NW = 0
               DO 80 KBND = 1, NBNDG2
                 IF (I1SW .EQ. IBNDG2(1,KBND)) K1SW = KBND
                 IF (I1NW .EQ. IBNDG2(1,KBND)) K1NW = KBND
                 IF (I2SW .EQ. IBNDG2(1,KBND)) K2SW = KBND
                 IF (I2NW .EQ. IBNDG2(1,KBND)) K2NW = KBND
80             CONTINUE
               IF (K1SW .NE. 0 .AND. K1NW .NE. 0) THEN
                 IF (K2SW .NE. 0 .AND. K2NW .NE. 0) WRITE(6,120)
C                IBNDG2(2,JBND) = IONE
                 IBNDG2(3,JBND) = 0
                 ICHAN          = ITWO
               ELSE IF (K2SW .NE. 0 .AND. K2NW .NE. 0) THEN
                 IBNDG2(2,JBND) = ITWO
                 IBNDG2(3,JBND) = 0
                 ICHAN          = IONE
               ELSE
                 WRITE(6,120)
               ENDIF

               WRITE(6,130) JBND, (IBNDG2(J,JBND),J=1,5)
C
C              THE AUXILLARY POINTERS OF SOME OF THE CELLS (ICHAN) PERTAINING
C              TO THE BOUNDARY NODES BEING CHANGED MAY HAVE TO BE RESET,
C              THE CELL WHOSE AUX. POINTER IS NON-ZERO AND YET IT DOES
```

```
C              NOT HAVE ANY ASSOCIATED POINTERS MUST HAVE THE "BOUNDARY
C              BYTE" OF THE AUX. POINTER AS ZERO
C           -
               KXO  = KAUXG2(ICHAN)
               KBXO = IAND(KXO,KLOOOF)
               IF (KBXO .EQ. 0) GOTO 90

C              RECOMMEND AN ALTERNATIVE POINTER
               KXN = IAND(KXO,KLFFFO)
               WRITE(6,150) ICHAN,KXO,ICHAN,KXN
               READ(5,160) KX

               IF (KX .GE. 0) THEN
                  KAUXG2(ICHAN) = KX
                  WRITE(6,170) ICHAN,KX
               ENDIF

            ENDIF
90       CONTINUE
100      FORMAT(/5X,'WANT TO CHANGE POINTERS OF THE BOUNDARY NODE')
120      FORMAT(5X,'G2PINJ: ERROR IN POINTERS: BOTH IONE AND ITWO HAVE',
     1           ' TWO NON-ZERO BOUNDARY NODES')
130      FORMAT(5X,'NEW BOUNDARY NODE POINTERS:',I5,5X,5I5)
150      FORMAT(//5X,'OLD AUXILLARY POINTER OF CELL   :',I5,5X,Z10/
     1            5X,'RECOMMENDED AUX. POINTER OF CELL:',I5,5X,Z10/
     2           10X,'INPUT NEW AUX. POINTER IN Z-FORMAT'/
     3               ' 12345678',5X,'INPUT -1 IF NO CHANGE IS DESIRED')
160      FORMAT(Z8)
170      FORMAT(5X,'NEW AUX. POINTER OF CELL:',I5,5X,Z10)

         RETURN
         END
```

GNREDN

```
         SUBROUTINE GNREDN

         INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
         INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
         INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
         INCLUDE 'GNBLOC.INC/LIST'

C*********************************************************************
C
C        THIS PROGRAM REMOVES REDUNDANCY OF NODES AT THE BOUNDARIES OF THE
C        VARIOUS ADJACENT BLOCKS
C
C*********************************************************************
C
C        FIRST INITIALIZE NODE KEEP ARRAY
C
         DO 10 INODE = 1, NNODG2
           PRESG2(INODE) = 1.0
10       CONTINUE
C
```

```
      DO 60 IBLOCK = 1, NBLOCK
C       EVALUATE THE SOUTHERN NEIGHBOUR BLOCK, IT EXISTS IF IT
C       IS POSITIVE, IN THIS CASE THE SOUTHERN SURFACE OF THE
C       CURRENT BLOCK WILL BE DELETED
        NBSURF = ISBLOCK(IBLOCK,1)
        IF (NBSURF .GT. 0) THEN
           DO 20 IX = 2, NXBLOCK(IBLOCK)-1
C             EVALUATE NODE TO BE DELETED
              INODED = IBS(IBLOCK,IX)
C             EVALUATE NODE TO BE KEPT
              INODEK = IBN(NBSURF,IX)
C             MARK THE NODE FOR DELETE
              PRESG2(INODED) = -1.0
C             RESET THE NEIGHBOUR NODE ARRAY
              NBCEL3 = NEIBG2(3,INODED)
              NBCEL4 = NEIBG2(4,INODED)
              NEIBG2(3,INODEK) = NBCEL3
              NEIBG2(4,INODEK) = NBCEL4
              IF (NBCEL3 .NE. 0) ICELG2(2,NBCEL3) = INODEK
              IF (NBCEL4 .NE. 0) ICELG2(4,NBCEL4) = INODEK
C             THE FOLLOWING IS NOT REALLY NEEDED, SINCE AFTER THIS
C             LOOP ONLY THE CORNER CELLS ARE USED
              IBS(IBLOCK,IX) = IBN(NBSURF,IX)
20         CONTINUE
        ENDIF
C
C       EVALUATE THE EASTERN NEIGHBOUR BLOCK
        NBSURF = ISBLOCK(IBLOCK,2)
        IF (NBSURF .GT. 0) THEN
           DO 30 IY = 2, NYBLOCK(IBLOCK)-1
C             EVALUATE NODE TO BE DELETED
              INODED = IBE(IBLOCK,IY)
C             EVALUATE NODE TO BE KEPT
              INODEK = IBW(NBSURF,IY)
C             MARK THE NODE FOR DELETE
              PRESG2(INODED) = -1.0
C             RESET THE NEIGHBOUR NODE ARRAY
              NBCEL1 = NEIBG2(1,INODED)
              NBCEL4 = NEIBG2(4,INODED)
              NEIBG2(1,INODEK) = NBCEL1
              NEIBG2(4,INODEK) = NBCEL4
              IF (NBCEL1 .NE. 0) ICELG2(6,NBCEL1) = INODEK
              IF (NBCEL4 .NE. 0) ICELG2(4,NBCEL4) = INODEK
              IBE(IBLOCK,IY) = IBW(NBSURF,IY)
30         CONTINUE
        ENDIF
C
C       EVALUATE THE NORTHERN NEIGHBOUR BLOCK
        NBSURF = ISBLOCK(IBLOCK,3)
        IF (NBSURF .GT. 0) THEN
           DO 40 IX = 2, NXBLOCK(IBLOCK)-1
C             EVALUATE NODE TO BE DELETED
              INODED = IBN(IBLOCK,IX)
C             EVALUATE NODE TO BE KEPT
              INODEK = IBS(NBSURF,IX)
C             MARK THE NODE FOR DELETE
              PRESG2(INODED) = -1.0
```

```
C              RESET THE NEIGHBOUR NODE ARRAY
               NBCEL1 = NEIBG2(1,INODED)
             - NBCEL2 = NEIBG2(2,INODED)
               NEIBG2(1,INODEK) = NBCEL1
               NEIBG2(2,INODEK) = NBCEL2
               IF (NBCEL1 .NE. 0) ICELG2(6,NBCEL1) = INODEK
               IF (NBCEL2 .NE. 0) ICELG2(8,NBCEL2) = INODEK
               IBN(IBLOCK,IX) = IBS(NBSURF,IX)
40          CONTINUE
          ENDIF
C
C         EVALUATE THE WESTERN NEIGHBOUR BLOCK
          NBSURF = ISBLOCK(IBLOCK,4)
          IF (NBSURF .GT. 0) THEN
             DO 50 IY = 2, NYBLOCK(IBLOCK)-1
C               EVALUATE NODE TO BE DELETED
                INODED = IBW(IBLOCK,IY)
C               EVALUATE NODE TO BE KEPT
                INODEK = IBE(NBSURF,IY)
C               MARK THE NODE FOR DELETE
                PRESG2(INODED) = -1.0
C               RESET THE NEIGHBOUR NODE ARRAY
                NBCEL2 = NEIBG2(2,INODED)
                NBCEL3 = NEIBG2(3,INODED)
                NEIBG2(2,INODEK) = NBCEL2
                NEIBG2(3,INODEK) = NBCEL3
                IF (NBCEL2 .NE. 0) ICELG2(8,NBCEL2) = INODEK
                IF (NBCEL3 .NE. 0) ICELG2(2,NBCEL3) = INODEK
                IBW(IBLOCK,IY) = IBE(NBSURF,IY)
50           CONTINUE
          ENDIF
C
60      CONTINUE
C
C       NOW MARK THE CORNER NODES
C
        DO 140 IBLOCK = 1, NBLOCK
C
C          SEE IF SW CORNER IS ALREADY DONE
C
           NCO = IBS(IBLOCK,1)
           IF (ISBLOCK(IBLOCK,1).GT.0 .OR. ISBLOCK(IBLOCK,4).GT.0)THEN
c             PRESG2(NCO) = -1.
              GOTO 80
           ENDIF
           XIH    = GEOMG2(1,NCO)
           YIH    = GEOMG2(2,NCO)

C          CHECK THE REST OF THE BLOCKS FOR SW CORNER
           DO 70 KBLOCK = IBLOCK+1, NBLOCK
              KCO = IBE(KBLOCK,1)
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL         = NEIBG2(4,KCO)
                  NEIBG2(4,NCO) = NCELL
                  IF (NCELL .NE. 0) ICELG2(4,NCELL) = NCO
```

```
                IBE(KBLOCK,1) = NCO
     -          IBS(KBLOCK,NXBLOCK(KBLOCK)) = NCO
                PRESG2(KCO) = -1.
              ENDIF
              KCO = IBN(KBLOCK,1)
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL         = NEIBG2(2,KCO)
                  NEIBG2(2,NCO) = NCELL
                  IF (NCELL .NE. 0) ICELG2(8,NCELL) = NCO
                  IBN(KBLOCK,1) = NCO
                  IBW(KBLOCK,NYBLOCK(KBLOCK)) = NCO
                  PRESG2(KCO) = -1.
              ENDIF
              KCO = IBN(KBLOCK,NXBLOCK(KBLOCK))
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL         = NEIBG2(1,KCO)
                  NEIBG2(1,NCO) = NCELL
                  IF (NCELL .NE. 0) ICELG2(6,NCELL) = NCO
                  IBN(KBLOCK,NXBLOCK(KBLOCK)) = NCO
                  IBE(KBLOCK,NYBLOCK(KBLOCK)) = NCO
                  PRESG2(KCO) = -1.
              ENDIF
70            CONTINUE
C
C         SEE IF SE CORNER IS ALREADY DONE
C
80        NCO = IBE(IBLOCK,1)
          IF (ISBLOCK(IBLOCK,1).GT.0 .OR. ISBLOCK(IBLOCK,2).GT.0)THEN
c             PRESG2(NCO) = -1.
              GOTO 100
          ENDIF
          XIH    = GEOMG2(1,NCO)
          YIH    = GEOMG2(2,NCO)

C         CHECK THE REST OF THE BLOCKS FOR SE CORNER
          DO 90 KBLOCK = IBLOCK+1, NBLOCK
              KCO = IBN(KBLOCK,1)
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL         = NEIBG2(2,KCO)
                  NEIBG2(2,NCO) = NCELL
                  IF (NCELL .NE. 0) ICELG2(8,NCELL) = NCO
                  IBN(KBLOCK,1) = NCO
                  IBW(KBLOCK,NYBLOCK(KBLOCK)) = NCO
                  PRESG2(KCO) = -1.
              ENDIF
              KCO = IBW(KBLOCK,1)
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL         = NEIBG2(3,KCO)
                  NEIBG2(3,NCO) = NCELL
```

```
                IF (NCELL .NE. 0) ICELG2(2,NCELL) = NCO
              -     IBW(KBLOCK,1) = NCO
                    IBS(KBLOCK,1) = NCO
                    PRESG2(KCO) = -1.
                ENDIF
                KCO = IBN(KBLOCK,NXBLOCK(KBLOCK))
                XKH = GEOMG2(1,KCO)
                YKH = GEOMG2(2,KCO)
                IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                    NCELL         = NEIBG2(1,KCO)
                    NEIBG2(1,NCO) = NCELL
                    IF (NCELL .NE. 0) ICELG2(6,NCELL) = NCO
                    IBN(KBLOCK,NXBLOCK(KBLOCK)) = NCO
                    IBE(KBLOCK,NYBLOCK(KBLOCK)) = NCO
                    PRESG2(KCO) = -1.
                ENDIF
90              CONTINUE
C
C           SEE IF NE CORNER IS ALREADY DONE
C
C           NCO = IBE(IBLOCK,NYBLOCK(IBLOCK))
100         NCO = IBN(IBLOCK,NXBLOCK(IBLOCK))
            IF (ISBLOCK(IBLOCK,2).GT.0 .OR. ISBLOCK(IBLOCK,3).GT.0)THEN
c               PRESG2(NCO) = -1.
                GOTO 120
            ENDIF
            XIH    = GEOMG2(1,NCO)
            YIH    = GEOMG2(2,NCO)

C           CHECK THE REST OF THE BLOCKS FOR NE CORNER
            DO 110 KBLOCK = IBLOCK+1, NBLOCK
                KCO = IBN(KBLOCK,1)
                XKH = GEOMG2(1,KCO)
                YKH = GEOMG2(2,KCO)
                IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                    NCELL         = NEIBG2(2,KCO)
                    NEIBG2(2,NCO) = NCELL
                    IF (NCELL .NE. 0) ICELG2(8,NCELL) = NCO
                    IBN(KBLOCK,1) = NCO
                    IBW(KBLOCK,NYBLOCK(KBLOCK)) = NCO
                    PRESG2(KCO) = -1.
                ENDIF
                KCO = IBS(KBLOCK,1)
                XKH = GEOMG2(1,KCO)
                YKH = GEOMG2(2,KCO)
                IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                    NCELL         = NEIBG2(3,KCO)
                    NEIBG2(3,NCO) = NCELL
                    IF (NCELL .NE. 0) ICELG2(2,NCELL) = NCO
                    IBS(KBLOCK,1) = NCO
                    IBW(KBLOCK,1) = NCO
                    PRESG2(KCO) = -1.
                ENDIF
                KCO = IBE(KBLOCK,1)
                XKH = GEOMG2(1,KCO)
                YKH = GEOMG2(2,KCO)
                IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
```

```
                NCELL          = NEIBG2(4,KCO)
     -          NEIBG2(4,NCO) = NCELL
                IF (NCELL .NE. 0) ICELG2(4,NCELL) = NCO
                IBE(KBLOCK,1) = NCO
                IBS(KBLOCK,NXBLOCK(KBLOCK)) = NCO
                PRESG2(KCO) = -1.
              ENDIF
110           CONTINUE
C
C          SEE IF NW CORNER IS ALREADY DONE
C
120        NCO = IBN(IBLOCK,1)
           IF (ISBLOCK(IBLOCK,3).GT.0 .OR. ISBLOCK(IBLOCK,4).GT.0)THEN
c             PRESG2(NCO) = -1.
              GOTO 140
           ENDIF
           XIH    = GEOMG2(1,NCO)
           YIH    = GEOMG2(2,NCO)

C          CHECK THE REST OF THE BLOCKS FOR NW CORNER
           DO 130 KBLOCK = IBLOCK+1, NBLOCK
              KCO = IBS(KBLOCK,1)
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL          = NEIBG2(3,KCO)
                  NEIBG2(3,NCO) = NCELL
                  IF (NCELL .NE. 0) ICELG2(2,NCELL) = NCO
                  IBS(KBLOCK,1) = NCO
                  IBW(KBLOCK,1) = NCO
                  PRESG2(KCO) = -1.
              ENDIF
              KCO = IBE(KBLOCK,1)
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL          = NEIBG2(4,KCO)
                  NEIBG2(4,NCO) = NCELL
                  IF (NCELL .NE. 0) ICELG2(4,NCELL) = NCO
                  IBE(KBLOCK,1) = NCO
                  IBS(KBLOCK,NXBLOCK(KBLOCK)) = NCO
                  PRESG2(KCO) = -1.
              ENDIF
              KCO = IBN(KBLOCK,NXBLOCK(KBLOCK))
              XKH = GEOMG2(1,KCO)
              YKH = GEOMG2(2,KCO)
              IF (XIH.EQ.XKH .AND. YIH.EQ.YKH) THEN
                  NCELL          = NEIBG2(1,KCO)
                  NEIBG2(1,NCO) = NCELL
                  IF (NCELL .NE. 0) ICELG2(6,NCELL) = NCO
                  IBN(KBLOCK,NXBLOCK(KBLOCK)) = NCO
                  IBE(KBLOCK,NYBLOCK(KBLOCK)) = NCO
                  PRESG2(KCO) = -1.
              ENDIF
130           CONTINUE

140     CONTINUE
```

```
C
C       DELETE THE NODES WHICH WERE MARKED EARLIER
C
        NNEW = 0
C
        DO 150 NOLD = 1, NNODG2
          IF (PRESG2(NOLD) .GT. 0.) THEN
             NNEW        = NNEW + 1
             KAUXG2(NOLD) = NNEW
             IF (NOLD .NE. NNEW) THEN
C               ADJUST THE GEOMETRY ARRAYS
                GEOMG2(1,NNEW) = GEOMG2(1,NOLD)
                GEOMG2(2,NNEW) = GEOMG2(2,NOLD)
C               ADJUST THE NEIGHBOUR-NODE-ARRAYS
                NEIBG2(1,NNEW) = NEIBG2(1,NOLD)
                NEIBG2(2,NNEW) = NEIBG2(2,NOLD)
                NEIBG2(3,NNEW) = NEIBG2(3,NOLD)
                NEIBG2(4,NNEW) = NEIBG2(4,NOLD)
             ENDIF
          ENDIF
150     CONTINUE

C       RESET NUMBER OF NODES

        NNODG2 = NNEW

C       STEP THROUGH ALL CELL POINTERS, WHICH POINT TOWARDS NODES,
C       REALIGNING TO NEW NODE NUMBERS.  THE NODE NUMBERS CORRESPONDING
C       TO COARSE CELLS ARE NOT CHANGED

        DO 170 ICELL = 1, NCELG2
C         STEP THROUGH EACH CELL POINTER
          DO 160 IPNT = 2, 8, 2
            IF (ICELG2(IPNT,ICELL) .NE. 0) THEN
                ICELG2(IPNT,ICELL) = KAUXG2(ICELG2(IPNT,ICELL))
            ENDIF
160       CONTINUE
170     CONTINUE

C       STEP THROUGH ALL THE BLOCK SURFACE POINTERS REALIGNING TO NEW
C       NODE NUMBERS.
C
        DO 200 IBLOCK = 1, NBLOCK

          IBS(IBLOCK,1) = KAUXG2(IBS(IBLOCK,1))
          IBE(IBLOCK,1) = KAUXG2(IBE(IBLOCK,1))
          IBN(IBLOCK,1) = KAUXG2(IBN(IBLOCK,1))
          IBW(IBLOCK,1) = KAUXG2(IBW(IBLOCK,1))
          IBS(IBLOCK,NXBLOCK(IBLOCK)) =
     1                    KAUXG2(IBS(IBLOCK,NXBLOCK(IBLOCK)))
          IBE(IBLOCK,NYBLOCK(IBLOCK)) =
     1                    KAUXG2(IBE(IBLOCK,NYBLOCK(IBLOCK)))
          IBN(IBLOCK,NXBLOCK(IBLOCK)) =
     1                    KAUXG2(IBN(IBLOCK,NXBLOCK(IBLOCK)))
          IBW(IBLOCK,NYBLOCK(IBLOCK)) =
     1                    KAUXG2(IBW(IBLOCK,NYBLOCK(IBLOCK)))
          DO 180 IX = 2, NXBLOCK(IBLOCK)-1
```

```
            IBS(IBLOCK,IX) = KAUXG2(IBS(IBLOCK,IX))
           -IBN(IBLOCK,IX) = KAUXG2(IBN(IBLOCK,IX))
180       CONTINUE

          DO 190 IY = 2, NYBLOCK(IBLOCK)-1
             IBW(IBLOCK,IY) = KAUXG2(IBW(IBLOCK,IY))
             IBE(IBLOCK,IY) = KAUXG2(IBE(IBLOCK,IY))
190       CONTINUE

200     CONTINUE
C
        RETURN
        END
```

GNSEPB

```
        SUBROUTINE GNSEPB (IBLOCK)

        INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
        INCLUDE 'GNBLOC.INC/LIST'
        DIMENSION DISTW(1000),   DISTE(1000)

C***********************************************************************
C
C       THIS SUBROUTINE IS IN-BOUNDARY-OUT-GRID (IBOG); I.E., IT TAKES
C       IN THE BOUNDARY INFORMATION AND GENERATES THE INTERIOR GRID
C
C***********************************************************************
C
C       COMPUTE THE NODE BEFORE THE FIRST NORTH ONE (L IN FIG.)
C       AND THE MAXIMUM NUMBER OF NODES

        NXRECT = NXBLOCK(IBLOCK)
        NYRECT = NYBLOCK(IBLOCK)
        NBEFNO = NXRECT*(NYRECT-1)
        NNODEH = NXRECT*NYRECT
C
C       SET SOUTH AND NORTH NODE INFORMATION

        DO 10 IX = 1, NXRECT
C          DETERMINE LOCAL NODES NOS AND NON; THE ACTUAL NODES ARE THESE
C          PLUS NNODG2 FROM PREVIOUS BLOCK
           NOS           = IX
           NON           = IX + NBEFNO
C          SAVE THE ACTUAL BOUNDARY NODES FOR THIS BLOCK
           IBS(IBLOCK,IX) = NOS + NNODG2
           IBN(IBLOCK,IX) = NON + NNODG2
C          SAVE THE GEOMETRY AT THE ACTUAL BOUNDARY NODES
           GEOMG2(1,NOS+NNODG2) = XSBLOCK(IBLOCK,IX)
           GEOMG2(2,NOS+NNODG2) = YSBLOCK(IBLOCK,IX)
           GEOMG2(1,NON+NNODG2) = XNBLOCK(IBLOCK,IX)
           GEOMG2(2,NON+NNODG2) = YNBLOCK(IBLOCK,IX)
10      CONTINUE
```

```
C
C       SET WEST AND EAST NODE INFORMATION

        DO 20 IY = 1, NYRECT
C          DETERMINE LOCAL NODES NOE AND NOW; THE ACTUAL NODES ARE THESE
C          PLUS NNODG2 FROM PREVIOUS BLOCK
           NOW           = 1 + (IY-1)*NXRECT
           NOE           = IY*NXRECT
C          SAVE THE ACTUAL BOUNDARY NODES FOR THIS BLOCK
           IBE(IBLOCK,IY) = NOE + NNODG2
           IBW(IBLOCK,IY) = NOW + NNODG2
C          SAVE THE GEOMETRY AT THE ACTUAL BOUNDARY NODES
           GEOMG2(1,NOW+NNODG2) = XWBLOCK(IBLOCK,IY)
           GEOMG2(2,NOW+NNODG2) = YWBLOCK(IBLOCK,IY)
           GEOMG2(1,NOE+NNODG2) = XEBLOCK(IBLOCK,IY)
           GEOMG2(2,NOE+NNODG2) = YEBLOCK(IBLOCK,IY)
20      CONTINUE
C
C       INITIALIZE THE FRACTIONAL DISTANCES ON WEST AND EAST EDGES
C
        DISTW(1) = 0.
        DISTE(1) = 0.
C
C       CALCULATE THE TOTAL DISTANCES ON WEST AND EAST EDGES
C
        DO 30 J = 2, NYRECT
           JM1      = J - 1
C
           INDJW    = 1 + (J  -1)*NXRECT
           INDJMW   = 1 + (JM1-1)*NXRECT
           INDJE    = J  *NXRECT
           INDJME   = JM1*NXRECT
C
           DXW      = GEOMG2(1,INDJW+NNODG2) - GEOMG2(1,INDJMW+NNODG2)
           DYW      = GEOMG2(2,INDJW+NNODG2) - GEOMG2(2,INDJMW+NNODG2)
           DXE      = GEOMG2(1,INDJE+NNODG2) - GEOMG2(1,INDJME+NNODG2)
           DYE      = GEOMG2(2,INDJE+NNODG2) - GEOMG2(2,INDJME+NNODG2)
C
           DISTW(J) = DISTW(JM1) + SQRT(DXW*DXW + DYW*DYW)
           DISTE(J) = DISTE(JM1) + SQRT(DXE*DXE + DYE*DYE)
C
30      CONTINUE

C       CALCULATE THE FRACTIONAL DISTANCES ON WEST AND EAST EDGES
C       FOR EACH NODE

        DO 40 J = 2, NYRECT
           DISTW(J) = DISTW(J)/DISTW(NYRECT)
           DISTE(J) = DISTE(J)/DISTE(NYRECT)
40      CONTINUE
C
C       STEP THROUGH EACH INTERIOR LINE
C
        DO 60 I = 2, NXRECT-1
           FRACI = FLOAT(I-1)/FLOAT(NXRECT-1)
C
C          CALCULATE FRACTIONAL DISTANCES FOR EACH INTERIOR POINT
```

```
C
         DO 50 J = 2,NYRECT-1
            FRACJ         = (1.-FRACI)*DISTW(J) + FRACI*DISTE(J)
C
            IND           = I + (     J-1)*NXRECT
            INDN          = I + (NYRECT-1)*NXRECT
            INDS          = I
C
C           COMPUTE THE DISTANCE FROM NORTH EDGE TO SOUTH EDGE
C
            DELXNS        = GEOMG2(1,INDN+NNODG2) - GEOMG2(1,INDS+NNODG2)
            DELYNS        = GEOMG2(2,INDN+NNODG2) - GEOMG2(2,INDS+NNODG2)
C
C           COMPUTE LOCATION OF INTERIOR POINT
C
            GEOMG2(1,IND+NNODG2) = GEOMG2(1,INDS+NNODG2) + FRACJ*DELXNS
            GEOMG2(2,IND+NNODG2) = GEOMG2(2,INDS+NNODG2) + FRACJ*DELYNS
C
50        CONTINUE
60      CONTINUE
C
C       COMPUTE NUMBER OF CELLS IN EACH DIRECTION ON THE GLOBAL MESH
C
        NXCELL = NXRECT - 1
        NYCELL = NYRECT - 1
C
C       LOOP THROUGH ALL GLOBAL GRID CELLS
C
C
        DO 70 JCELL = 1, NYCELL
          DO 70 ICELL = 1, NXCELL
C
            NCELG2 = NCELG2 + 1
C
C           COMPUTE INDICES OF CORNER OF CELL
C
            ICELG2(2,NCELG2) = ICELL     + (JCELL-1)*NXRECT + NNODG2
            ICELG2(4,NCELG2) = ICELL + 1 + (JCELL-1)*NXRECT + NNODG2
            ICELG2(6,NCELG2) = ICELL + 1 + (JCELL  )*NXRECT + NNODG2
            ICELG2(8,NCELG2) = ICELL     + (JCELL  )*NXRECT + NNODG2
C
C           INITIALLY, THERE IS NO NODE IN THE CENTER OF A FINE CELL
C
            ICELG2(1,NCELG2) = 0
C
C           THERE ARE NO NODES IN THE CENTER OF THE SIDES OF A FINE CELL
C
            ICELG2(3 ,NCELG2) = 0
            ICELG2(5 ,NCELG2) = 0
            ICELG2(7 ,NCELG2) = 0
            ICELG2(9 ,NCELG2) = 0
            ICELG2(10,NCELG2) = 0
C
70      CONTINUE
C
C       SET UP NEIGHBOUR-NODE-ARRAY
C
```

```
      DO 80 LCELL = 1, NCELG2
         KSW           = ICELG2(2,LCELL)
         KSE           = ICELG2(4,LCELL)
         KNE           = ICELG2(6,LCELL)
         KNW           = ICELG2(8,LCELL)
         NEIBG2(1,KNE) = LCELL
         NEIBG2(2,KNW) = LCELL
         NEIBG2(3,KSW) = LCELL
         NEIBG2(4,KSE) = LCELL
80    CONTINUE
C
C          ------------
C          NOMENCLATURE
C          ------------
C                                                        NX=NXRECT
C                                                        NY=NYRECT
C                       L  L  L                          L =NBEFNO
C                       +  +  + . . .
C                       1  2  3
C           1+(NY-1)*NX +--+--+--+--+--+--+--+ NY*NX
C                       +       NORTH        + (NY-1)*NX = L
C                       + W                E +
C  INDJW                + E                A +                  INDJE
C                   ... + S                S + ...
C                1+2*NX + T                T + 3*NX
C                  1+NX +       SOUTH        + 2*NX
C                     1 +--+--+--+--+--+--+--+ NX
C                          2  3  ...     NX-1
C
C
      RETURN
      END
```

GNWEDG

```
      SUBROUTINE GNWEDG

      INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] HEXCOD.INC     '
      DIMENSION MARKBN(MBNDG2)
      CHARACTER*1 YESNO

C*********************************************************************
C
C     THIS SUBROUTINE IS USEFUL TO REMOVE THE EXTRA CORNER BOUNDARY
C     NODES WHEN THERE ARE EMBEDDED WEDGES OR DIAMONDS IN THE FLOW-
C     FIELD.  THESE CORNER POINTS ARE REGARDED AS THE REGULAR POINTS,
C     FOR EXAMPLE, IN THE "KUMAR" CASE AND IN THE "PARALLEL INJECTOR"
C     CASE.
C
C*********************************************************************
C
C     INITIALIZE THE NUMBER OF INTERIOR POINTS
```

```
C
      KOUNTI = 0
C
C     LOCATE ALL THE INTERIOR CORNER NODES.  THESE NODES ARE DEFINED
C     AS THE NORTHERN OR SOUTHERN SLIP-BOUNDARIES IN THE INITIAL GRIDS

      DO 10 JBND = 1, NBNDG2

         INODE  = IBNDG2(1,JBND)
         IEDGE  = IBNDG2(4,JBND)
         IBCTYP = IBNDG2(5,JBND)

C        CHECK SOUTHERN SLIP BOUNDARY
C        THIS WOULD BE A REAL SOUTHERN BOUNDARY IF THE SOUTHERN
C        NEIGHBOUR CELLS OF THE NODE "INODE" ARE NOT DEFINED,
C        ELSE IT WOULD AN INTERIOR NODE

         IF (IEDGE .EQ. 3 .AND. IBCTYP .EQ. 3) THEN
            NBSW = NEIBG2(1,INODE)
            NBSE = NEIBG2(2,INODE)
            IF (NBSW .NE. 0 .AND. NBSE .NE. 0) THEN
               KOUNTI = KOUNTI + 1
               MARKBN(KOUNTI) = JBND
            ENDIF
         ENDIF

C        NOW CHECK NORTHERN SLIP BOUNDARY
C        THIS WOULD BE A REAL NORTHERN BOUNDARY IF THE NORTHERN
C        NEIGHBOUR CELLS OF THE NODE "INODE" ARE NOT DEFINED,
C        ELSE IT WOULD AN INTERIOR NODE

         IF (IEDGE .EQ. 7 .AND. IBCTYP .EQ. 3) THEN
            NBNE = NEIBG2(3,INODE)
            NBNW = NEIBG2(4,INODE)
            IF (NBNE .NE. 0 .AND. NBNW .NE. 0) THEN
               KOUNTI = KOUNTI + 1
               MARKBN(KOUNTI) = JBND
            ENDIF
         ENDIF

10    CONTINUE

C     WRITE DOWN ALL THE INTERIOR CORNER BOUNDARY NODES AND QUERY IF
C     ANY OF THEM HAVE TO BE DELETED

      IF (KOUNTI .EQ. 0) GOTO 135
      WRITE(6,20)
20    FORMAT(/5X,'THE FOLLOWING INTERIOR CORNER BOUNDARY NODES',
     1           ' ARE FOUND'/)

      DO 30 KOUNT = 1, KOUNTI
         JBND = MARKBN(KOUNT)
         WRITE(6,40) JBND, (IBNDG2(J,JBND),J=1,5)
30    CONTINUE
40    FORMAT(7X,'BOUNDARY  NODE:',I5,5X,5I5)

      WRITE(6,50)
```

```
50      FORMAT(/5X,'WANT TO DELETE ANY OF THE CORNER BOUNDARY NODES')
        -
        READ (5,60) YESNO
60      FORMAT(A)

        IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y') GOTO 70
        GOTO 135
C
C       INITIALIZE THE NUMBER OF INTERIOR BOUNDARY NODES TO BE DELETED
C
70      KOUNTD = 0
C
C       SEE IF ANY OF THE CORNER BOUNDARY NODES ARE TO BE DELETED
C
        DO 80 KOUNT = 1, KOUNTI
           JBND = MARKBN(KOUNT)
           WRITE(6,90)
           WRITE(6,40) JBND, (IBNDG2(J,JBND),J=1,5)
           READ (5,60) YESNO
           IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y') THEN
               KOUNTD = KOUNTD + 1
               IBNDG2(1,JBND) = -9
           ENDIF
80      CONTINUE
90      FORMAT(/5X,'WANT TO DELETE THE CORNER BOUNDARY NODE OF')
C
C       IF ANY OF THE CORNER BOUNARY NODES ARE REALLY DELETED THEN
C       THE REST OF THE BOUNDARY POINTERS HAVE TO BE RE-ALLIGNED
C
        IF (KOUNTD .EQ. 0) GOTO 135

C       SET THE COUNTER FOR ALL BOUNDARY NODES TO KEEP
        NNEW = 0

        DO 110 NOLD = 1, NBNDG2

            MARKBN(NOLD) = 0
C
C           MARK THE "KEEP" BOUNDARY NODES AND DELETE ALL BOUNDARY
C           CONDITION POINTERS MARKED FOR DELETE

            IF (IBNDG2(1,NOLD) .NE. -9) THEN
C
               NNEW = NNEW + 1
               MARKBN(NOLD) = NNEW
C              MOVE POINTER INFORMATION
               IF (NOLD .NE. NNEW) THEN
                  DO 100 J = 1, 5
                     IBNDG2(J,NNEW) = IBNDG2(J,NOLD)
100               CONTINUE
               ENDIF
            ENDIF
110     CONTINUE

C       RESET NUMBER OF BOUNDARY CONDITION POINTERS
C
        NBNDG2 = NNEW
```

```
      DO 130 IEDGE = 1, 4
         DO 120 IBND = 1, 2
            NBCPG2(IEDGE,IBND) = MARKBN(NBCPG2(IEDGE,IBND))
120      CONTINUE
130   CONTINUE
      WRITE(6,*) ' NCELG2=',NCELG2
      WRITE(6,*) ' NNODG2=',NNODG2
      WRITE(6,*) ' NBNDG2=',NBNDG2
C
C     THE AUXILLARY POINTERS OF SOME OF THE CELLS PERTAINING
C     TO THE BOUNDARY NODES BEING DELETED MAY HAVE TO BE RESET,
C     THE CELL WHOSE AUX. POINTER IS NON-ZERO AND YET IT DOES
C     NOT HAVE ANY ASSOCIATED POINTERS MUST HAVE THE "BOUNDARY
C     BYTE" OF THE AUX. POINTER AS ZERO
C
135   DO 145 ICELL = 1, NCELG2
          IP1  = ICELG2(2,ICELL)
          IP2  = ICELG2(4,ICELL)
          IP3  = ICELG2(6,ICELL)
          IP4  = ICELG2(8,ICELL)
          KXO  = KAUXG2(ICELL)
          KBXO = IAND(KXO,KLOOOF)
          IF (KBXO .EQ. 0) GOTO 145

          DO 140 KBND = 1, NBNDG2
            IF (IP1 .EQ. IBNDG2(1,KBND)) GOTO 145
            IF (IP2 .EQ. IBNDG2(1,KBND)) GOTO 145
            IF (IP3 .EQ. IBNDG2(1,KBND)) GOTO 145
            IF (IP4 .EQ. IBNDG2(1,KBND)) GOTO 145
140       CONTINUE

C         NONE OF THE BOUNDARY NODES FOR THIS CELL EXIST, SO
C         RECOMMEND AN ALTERNATIVE POINTER

          KXN = IAND(KXO,KLFFFO)
          WRITE(6,150) ICELL,KXO,ICELL,KXN
          READ(5,160) KX
          IF (KX .GE. 0) THEN
             KAUXG2(ICELL) = KX
             WRITE(6,170) ICELL,KX
          ENDIF

145   CONTINUE

150   FORMAT(//5X,'OLD AUXILLARY POINTER OF CELL   :',I5,5X,Z10/
     1         5X,'RECOMMENDED AUX. POINTER OF CELL:',I5,5X,Z10/
     2        10X,'INPUT NEW AUX. POINTER IN Z-FORMAT'/
     3            ' 12345678',5X,'INPUT -1 IF NO CHANGE IS DESIRED')
160   FORMAT(Z8)
170   FORMAT(5X,'NEW AUX. POINTER OF CELL:',I5,5X,Z10)

      RETURN
      END
```

ZRGNBN

```
      SUBROUTINE ZRGNBN (IFUN, INDGR, PLTITL, ALIMITS,ISTRING,
     1    A1, A2, A3, A4, A5, A6, A7, A8, A9, A10)

      DIMENSION ALIMITS(*)
      INCLUDE 'GNBLOC.INC/LIST'

      JSYM = NINT(A1)
      GOTO (1000,2000,1000,4000), IFUN+1
1000  RETURN
2000  CONTINUE

      XMIN = 1000.
      XMAX =-1000.
      YMIN = 1000.
      YMAX =-1000.

      DO 2030 IBLOCK = 1, NBLOCK
         DO 2010 IY = 1, NYBLOCK(IBLOCK)
            YMIN = MIN (YMIN,YSBLOCK(IBLOCK,IY))
            YMAX = MAX (YMAX,YNBLOCK(IBLOCK,IY))
2010     CONTINUE

         DO 2020 IX = 1, NXBLOCK(IBLOCK)
            XMIN = MIN (XMIN,XWBLOCK(IBLOCK,IX))
            XMAX = MAX (XMAX,XEBLOCK(IBLOCK,IX))
2020     CONTINUE

2030  CONTINUE

      ALIMITS(1) = XMIN
      ALIMITS(2) = XMAX
      ALIMITS(3) = YMIN
      ALIMITS(4) = YMAX
      RETURN

4000  CONTINUE
      if (jsym .eq. 0) goto 4500

      DO 4050 IBLOCK = 1, NBLOCK
         CALL GR_MOVE (XSBLOCK(IBLOCK,1),YSBLOCK(IBLOCK,1),JSYM )
         DO 4010 IX = 2, NXBLOCK(IBLOCK)
           CALL GR_DRAW (XSBLOCK(IBLOCK,IX),YSBLOCK(IBLOCK,IX),JSYM )
4010     CONTINUE
         DO 4020 IY = 2, NYBLOCK(IBLOCK)
           CALL GR_DRAW (XEBLOCK(IBLOCK,IY),YEBLOCK(IBLOCK,IY),JSYM )
4020     CONTINUE

         CALL GR_MOVE (XWBLOCK(IBLOCK,1),YWBLOCK(IBLOCK,1),JSYM )
         DO 4030 IY = 2, NYBLOCK(IBLOCK)
           CALL GR_DRAW (XWBLOCK(IBLOCK,IY),YWBLOCK(IBLOCK,IY),JSYM )
4030     CONTINUE
         DO 4040 IX = 2, NXBLOCK(IBLOCK)
           CALL GR_DRAW (XNBLOCK(IBLOCK,IX),YNBLOCK(IBLOCK,IX),JSYM )
4040     CONTINUE
```

```
4050    CONTINUE

        RETURN

4500    DO 4550 IBLOCK = 1, NBLOCK
           CALL GR_MOVE (XSBLOCK(IBLOCK,1),YSBLOCK(IBLOCK,1),JSYM )
           if (isblock(iblock,1).ge.0) goto 4511
           DO 4510 IX = 2, NXBLOCK(IBLOCK)
             CALL GR_DRAW (XSBLOCK(IBLOCK,IX),YSBLOCK(IBLOCK,IX),JSYM )
4510       CONTINUE
4511       CALL GR_MOVE (XeBLOCK(IBLOCK,1),YeBLOCK(IBLOCK,1),JSYM )
           if (isblock(iblock,2).ge.0) goto 4521
           DO 4520 IY = 2, NYBLOCK(IBLOCK)
             CALL GR_DRAW (XEBLOCK(IBLOCK,IY),YEBLOCK(IBLOCK,IY),JSYM )
4520       CONTINUE
4521       CALL GR_MOVE (XWBLOCK(IBLOCK,1),YWBLOCK(IBLOCK,1),JSYM )
           if (isblock(iblock,4).ge.0) goto 4531
           DO 4530 IY = 2, NYBLOCK(IBLOCK)
             CALL GR_DRAW (XWBLOCK(IBLOCK,IY),YWBLOCK(IBLOCK,IY),JSYM )
4530       CONTINUE
4531       CALL GR_MOVE (XnBLOCK(IBLOCK,1),YnBLOCK(IBLOCK,1),JSYM )
           if (isblock(iblock,3).ge.0) goto 4540
           DO 4540 IX = 2, NXBLOCK(IBLOCK)
             CALL GR_DRAW (XNBLOCK(IBLOCK,IX),YNBLOCK(IBLOCK,IX),JSYM )
4540       CONTINUE

4550    CONTINUE

        RETURN
        END
```

D.3 STAR Code

This section contains information on the spatio-temporal algorithm STAR.

D.3.1 Common Files

The file ALLINC.INC includes declaration and common block statements. Portions of this file are to be included with the appropriate INCLUDE statements in the following FORTRAN code listing.

```
C       PRECIS.INC
C       IMPLICIT REAL*4 (A-G,O-Y)


C       PARMV2.INC
        PARAMETER (    MEQNFL = 10    , MREACH = 20   , MSPECH = 6    ,
     1                 MNODG2 =16000 , MCELG2 =20000, MBNDG2 = 1000,
     2                 MLVLG2 = 5     , NIPAKY = 42   , NAPAKY = 42   ,
     3                 MMAXTI = 6                                      )
        COMMON/MNCOMN/ NEQNFL, NREACH, NSPECH, NNODG2, NCELG2, NBNDG2,
     1                 NLVLG2, NEQBAS, KROGER, MTITLE
        CHARACTER*80   MTITLE


C       A2COMN.INC
        COMMON/A2COMN/ NXTDA2, METHA2, NCELA2, K1ADA2, K2ADA2, MTYPA2,
     1         NPLCA2, IDBGA2, MITRA2, KCHKA2, MTHRA2, KPLTA2, KMERA2,
     2         NHNGA2, NNODA2, ILVLA2(2,0:MMAXTI),
     3         ICELA2(MCELG2), MRKCA2(MCELG2), MRKDA2(MCELG2),
     4         ALPHA2, BETAA2, GAMMA2, DELTA2, THRDA2, THRCA2,
     5         CHNGA2(MCELG2), WORKA2(MNODG2)


C       CHCOMN.INC
        COMMON/CHCOMN/ PREECH(MREACH), EXPECH(MREACH), ENEECH(MREACH),
     1                 PREFCH(MREACH), EXPFCH(MREACH), ENEFCH(MREACH),
     2         TREFCH, PREBCH(MREACH), EXPBCH(MREACH), ENEBCH(MREACH),
     3         PRESCH, FMHTCH(MSPECH), SPCPCH(MSPECH), SPCVCH(MSPECH),
     4         TRIGCH, ENTRCH(MSPECH), YSPECH(MSPECH), AMWTCH(MSPECH),
     5         YNRTCH, SPBSCH(MSPECH), YMAXCH(MSPECH),
     6                 RAMWCH(MSPECH), BMIACH(MSPECH,MREACH)          ,
     7         IDBGCH, IALPCH(MSPECH,MREACH), IBETCH(MSPECH,MREACH)   ,
     8         NINRCH, IALOCH(MSPECH,MREACH), IBTOCH(MSPECH,MREACH)   ,
     9         NEQSCH, NSRKCH(MREACH)       , ITABCH(MSPECH,MREACH)


C       E2COMN.INC
        COMMON/E2COMN/ IDBGE2,  MITRE2,  KSRTE2,  NITRE2       ,
     1                 KONVE2,  KEQNE2,  SIGGE2(MNODG2)        ,
```

```
     2                ERORE2,  EPSLE2,  CHNGE2(MEQNFL,MNODG2),
     3      -         SDELE2,  SMAXE2,  SMINE2,  RREYE2      ,
     4                RPRNE2,  RSCHE2,  OMEGE2,  GFACE2


C       FLCOMN.INC
        COMMON/FLCOMN/ TREFFL, PRESFL, UGASFL, AMCHFL, DISTFL, RHORFL,
     1                UREFFL, FMREFL, WDREFL, AMWTFL, GAMAFL, IDBGFL


C       FRCOMN.INC
        COMMON/FRCOMN/ RHORFR, UCOMFR, VCOMFR, PRESFR, PBPIFR,
     1         DPENFR(MEQNFL), IDBGFR, KPERFR, MCYCFR, NCYCFR


C       G2COMN.INC
        COMMON/G2COMN/ DPENG2(MEQNFL,MNODG2), TEMPG2(MNODG2),
     1                 GEOMG2(2     ,MNODG2), PRESG2(MNODG2),
     2         IDBGG2, KAUXG2(  MCELG2), ILVLG2(3     ,-MLVLG2:MLVLG2),
     3         MALVG2, NEIBG2(4,MNODG2), ICELG2(10    , MCELG2       ),
     4         NCRSG2, IBNDG2(5,MBNDG2), NBCPG2(4,2)


C       H2COMN.INC
        PARAMETER (MUMDH2=100)
        COMMON/H2COMN/ IADDH2, NODEH2(MUMDH2), NUMDH2, PHIEH2,
     1                 NCELH2, ICELH2(MUMDH2), IBASH2


C       IOCOMN.INC
        COMMON/IOCOMN/ JCARDS, JDEBUG, JDUMY1, JDUMY2, JDUMY3, JDUMY4,
     1         JHISTO, JOUTAL, JPNTRE, JPNTWR, JPRINT, JREADC, JREADD,
     2         JREADF, JREADG, JREADI, JREADS, JTERMI, JTERMO


C       JACOMN.INC
        COMMON/JACOMN/  BGF2JA, BGF4JA, BGG3JA, BGG4JA,
     1                  BIGWJA(MEQNFL), DPENJA(MEQNFL)


C       KYCOMN.INC
        COMMON/KYCOMN/  IPASKY(NIPAKY) ,   APASKY(NAPAKY),
     1                  MARIKY(NIPAKY) ,   MARAKY(NAPAKY)


C       M2COMN.INC
        COMMON/M2COMN/ RVOLM2(MCELG2), PERIM2(MCELG2), DXEWM2(MCELG2),
     1                 DYEWM2(MCELG2), DXNSM2(MCELG2), DYNSM2(MCELG2)


C       PRCOMN.INC
        COMMON/PRCOMN/ AMCHPR, BEPSPR, GAMAPR, PRESPR, RHORPR, SONDPR,
     1                 TEMPPR, UCOMPR, VCOMPR, YSPEPR(MSPECH)


C       SPCOMN.INC
        COMMON/SPCOMN/ LBNDG2, JBNDG2(3,10)
```

```
C       TICOMN.INC
        COMMON/TICOMN/ CFLNTI, CFLXTI, DTMNTI, EPSOTI, EPS1TI, TIMXTI,
     1                 TIMNTI, DTCNTI, FCTRTI, ERRMTI, CELLTI(MCELG2),
     2                 IMPLTI, KADPTI, KDIFTI, KTIMTI, NGIVTI, NMAXTI,
     3                 KFACTI, ICELTI(MCELG2), ILVLTI(2,0:MMAXTI)


C       TVCOMN.INC
        PARAMETER (MUMNTV=100)
        COMMON/TVCOMN/ AMPLTV, FLOWTV, FREQTV, NODETV(MUMNTV), NUMNTV


C       HEXCOD.INC
        DATA KL0000 /Z00000000/, KL0001 /Z00000001/, KL0002 /Z00000002/,
     1       KL0003 /Z00000003/, KL0004 /Z00000004/, KL0005 /Z00000005/,
     2       KL0006 /Z00000006/, KL0007 /Z00000007/, KL0008 /Z00000008/,
     3       KL0009 /Z00000009/, KL000A /Z0000000A/, KL000B /Z0000000B/,
     4       KL000C /Z0000000C/, KL000D /Z0000000D/, KL000E /Z0000000E/,
     5       KL000F /Z0000000F/
        DATA KL0010 /Z00000010/, KL0020 /Z00000020/, KL0030 /Z00000030/,
     1       KL0040 /Z00000040/, KL0050 /Z00000050/, KL0060 /Z00000060/,
     2       KL0070 /Z00000070/, KL0080 /Z00000080/, KL0090 /Z00000090/,
     3       KL00A0 /Z000000A0/, KL00B0 /Z000000B0/, KL00C0 /Z000000C0/,
     4       KL00D0 /Z000000D0/, KL00E0 /Z000000E0/, KL00F0 /Z000000F0/
        DATA KL0100 /Z00000100/, KL0200 /Z00000200/, KL0300 /Z00000300/,
     1       KL0400 /Z00000400/, KL0500 /Z00000500/, KL0600 /Z00000600/,
     2       KL0700 /Z00000700/, KL0800 /Z00000800/, KL0900 /Z00000900/,
     3       KL0A00 /Z00000A00/, KL0B00 /Z00000B00/, KL0C00 /Z00000C00/,
     4       KL0D00 /Z00000D00/, KL0E00 /Z00000E00/, KL0F00 /Z00000F00/
        DATA KL1000 /Z00001000/, KL2000 /Z00002000/, KL3000 /Z00003000/,
     1       KL4000 /Z00004000/, KL5000 /Z00005000/, KL6000 /Z00006000/,
     2       KL7000 /Z00007000/, KL8000 /Z00008000/, KL9000 /Z00009000/,
     3       KLA000 /Z0000A000/, KLB000 /Z0000B000/, KLC000 /Z0000C000/,
     4       KLD000 /Z0000D000/, KLE000 /Z0000E000/, KLF000 /Z0000F000/
        DATA KU0000 /Z00000000/, KU0001 /Z00010000/, KU0002 /Z00020000/,
     1       KU0003 /Z00030000/, KU0004 /Z00040000/, KU0005 /Z00050000/,
     2       KU0006 /Z00060000/, KU0007 /Z00070000/, KU0008 /Z00080000/,
     3       KU0009 /Z00090000/, KU000A /Z000A0000/, KU000B /Z000B0000/,
     4       KU000C /Z000C0000/, KU000D /Z000D0000/, KU000E /Z000E0000/,
     5       KU000F /Z000F0000/
        DATA KU0010 /Z00100000/, KU0020 /Z00200000/, KU0030 /Z00300000/,
     1       KU0040 /Z00400000/, KU0050 /Z00500000/, KU0060 /Z00600000/,
     2       KU0070 /Z00700000/, KU0080 /Z00800000/, KU0090 /Z00900000/,
     3       KU00A0 /Z00A00000/, KU00B0 /Z00B00000/, KU00C0 /Z00C00000/,
     4       KU00D0 /Z00D00000/, KU00E0 /Z00E00000/, KU00F0 /Z00F00000/
        DATA KU0100 /Z01000000/, KU0200 /Z02000000/, KU0300 /Z03000000/,
     1       KU0400 /Z04000000/, KU0500 /Z05000000/, KU0600 /Z06000000/,
     2       KU0700 /Z07000000/, KU0800 /Z08000000/, KU0900 /Z09000000/,
     3       KU0A00 /Z0A000000/, KU0B00 /Z0B000000/, KU0C00 /Z0C000000/,
     4       KU0D00 /Z0D000000/, KU0E00 /Z0E000000/, KU0F00 /Z0F000000/
        DATA KU1000 /Z10000000/, KU2000 /Z20000000/, KU3000 /Z30000000/,
     1       KU4000 /Z40000000/, KU5000 /Z50000000/, KU6000 /Z60000000/,
     2       KU7000 /Z70000000/, KU8000 /Z80000000/, KU9000 /Z90000000/,
     3       KUA000 /ZA0000000/, KUB000 /ZB0000000/, KUC000 /ZC0000000/,
     4       KUD000 /ZD0000000/, KUE000 /ZE0000000/, KUF000 /ZF0000000/
```

```
      DATA KLFFF0 /ZFFFFFFF0/, KLFFF1 /ZFFFFFFF1/, KLFFF2 /ZFFFFFFF2/,
     1      -KLFFF3 /ZFFFFFFF3/, KLFFF4 /ZFFFFFFF4/, KLFFF5 /ZFFFFFFF5/,
     2       KLFFF6 /ZFFFFFFF6/, KLFFF7 /ZFFFFFFF7/, KLFFF8 /ZFFFFFFF8/,
     3       KLFFF9 /ZFFFFFFF9/, KLFFFA /ZFFFFFFFA/, KLFFFB /ZFFFFFFFB/,
     4       KLFFFC /ZFFFFFFFC/, KLFFFD /ZFFFFFFFD/, KLFFFE /ZFFFFFFFE/,
     5       KLFFFF /ZFFFFFFFF/
      DATA KLFF0F /ZFFFFFF0F/, KLFF1F /ZFFFFFF1F/, KLFF2F /ZFFFFFF2F/,
     1       KLFF3F /ZFFFFFF3F/, KLFF4F /ZFFFFFF4F/, KLFF5F /ZFFFFFF5F/,
     2       KLFF6F /ZFFFFFF6F/, KLFF7F /ZFFFFFF7F/, KLFF8F /ZFFFFFF8F/,
     3       KLFF9F /ZFFFFFF9F/, KLFFAF /ZFFFFFFAF/, KLFFBF /ZFFFFFFBF/,
     4       KLFFCF /ZFFFFFFCF/, KLFFDF /ZFFFFFFDF/, KLFFEF /ZFFFFFFEF/
      DATA KLF0FF /ZFFFFF0FF/, KLF1FF /ZFFFFF1FF/, KLF2FF /ZFFFFF2FF/,
     1       KLF3FF /ZFFFFF3FF/, KLF4FF /ZFFFFF4FF/, KLF5FF /ZFFFFF5FF/,
     2       KLF6FF /ZFFFFF6FF/, KLF7FF /ZFFFFF7FF/, KLF8FF /ZFFFFF8FF/,
     3       KLF9FF /ZFFFFF9FF/, KLFAFF /ZFFFFFAFF/, KLFBFF /ZFFFFFBFF/,
     4       KLFCFF /ZFFFFFCFF/, KLFDFF /ZFFFFFDFF/, KLFEFF /ZFFFFFEFF/
      DATA KL0FFF /ZFFFF0FFF/, KL1FFF /ZFFFF1FFF/, KL2FFF /ZFFFF2FFF/,
     1       KL3FFF /ZFFFF3FFF/, KL4FFF /ZFFFF4FFF/, KL5FFF /ZFFFF5FFF/,
     2       KL6FFF /ZFFFF6FFF/, KL7FFF /ZFFFF7FFF/, KL8FFF /ZFFFF8FFF/,
     3       KL9FFF /ZFFFF9FFF/, KLAFFF /ZFFFFAFFF/, KLBFFF /ZFFFFBFFF/,
     4       KLCFFF /ZFFFFCFFF/, KLDFFF /ZFFFFDFFF/, KLEFFF /ZFFFFEFFF/
      DATA KUFFF0 /ZFFF0FFFF/, KUFFF1 /ZFFF1FFFF/, KUFFF2 /ZFFF2FFFF/,
     1       KUFFF3 /ZFFF3FFFF/, KUFFF4 /ZFFF4FFFF/, KUFFF5 /ZFFF5FFFF/,
     2       KUFFF6 /ZFFF6FFFF/, KUFFF7 /ZFFF7FFFF/, KUFFF8 /ZFFF8FFFF/,
     3       KUFFF9 /ZFFF9FFFF/, KUFFFA /ZFFFAFFFF/, KUFFFB /ZFFFBFFFF/,
     4       KUFFFC /ZFFFCFFFF/, KUFFFD /ZFFFDFFFF/, KUFFFE /ZFFFEFFFF/,
     5       KUFFFF /ZFFFFFFFF/
      DATA KUFF0F /ZFF0FFFFF/, KUFF1F /ZFF1FFFFF/, KUFF2F /ZFF2FFFFF/,
     1       KUFF3F /ZFF3FFFFF/, KUFF4F /ZFF4FFFFF/, KUFF5F /ZFF5FFFFF/,
     2       KUFF6F /ZFF6FFFFF/, KUFF7F /ZFF7FFFFF/, KUFF8F /ZFF8FFFFF/,
     3       KUFF9F /ZFF9FFFFF/, KUFFAF /ZFFAFFFFF/, KUFFBF /ZFFBFFFFF/,
     4       KUFFCF /ZFFCFFFFF/, KUFFDF /ZFFDFFFFF/, KUFFEF /ZFFEFFFFF/
      DATA KUF0FF /ZF0FFFFFF/, KUF1FF /ZF1FFFFFF/, KUF2FF /ZF2FFFFFF/,
     1       KUF3FF /ZF3FFFFFF/, KUF4FF /ZF4FFFFFF/, KUF5FF /ZF5FFFFFF/,
     2       KUF6FF /ZF6FFFFFF/, KUF7FF /ZF7FFFFFF/, KUF8FF /ZF8FFFFFF/,
     3       KUF9FF /ZF9FFFFFF/, KUFAFF /ZFAFFFFFF/, KUFBFF /ZFBFFFFFF/,
     4       KUFCFF /ZFCFFFFFF/, KUFDFF /ZFDFFFFFF/, KUFEFF /ZFEFFFFFF/
      DATA KU0FFF /Z0FFFFFFF/, KU1FFF /Z1FFFFFFF/, KU2FFF /Z2FFFFFFF/,
     1       KU3FFF /Z3FFFFFFF/, KU4FFF /Z4FFFFFFF/, KU5FFF /Z5FFFFFFF/,
     2       KU6FFF /Z6FFFFFFF/, KU7FFF /Z7FFFFFFF/, KU8FFF /Z8FFFFFFF/,
     3       KU9FFF /Z9FFFFFFF/, KUAFFF /ZAFFFFFFF/, KUBFFF /ZBFFFFFFF/,
     4       KUCFFF /ZCFFFFFFF/, KUDFFF /ZDFFFFFFF/, KUEFFF /ZEFFFFFFF/
```

D.3.2 Link information

The file STAR.COM contains link information for the STAR code.

```
$ LINK/EXE=STAR    TWODOU,   A2ADUU,   A2CEWC,   A2EXTU,   A2GRDC,-
        A2GRDN,   A2INIT,   A2MDFU,   A2MTHU,   A2THRU,   A2VALC,-
        A2VALN,   A2VOUU,   C2EQDI,   C2HELP,   C2INIT,   C2KCRE,-
        C2PONT,   C2RINT,   C2ROCH,   CHKBN2,   CHKMAS,   CHKNC2,-
        CHKNN2,   CHKPR2,   CHKREF,   CHKSP2,   CHKTM2,   CHKYMX,-
```

```
DPINI2,   E2BCNF,   E2CONO,   E2DIFF,   E2FINI,   E2INIO,-
E2OPTO,   E2PRMU,   E2RSRT,   E2SCHO,   E2SOLF,   E2SOUU,-
E2TIMC,   E2TIMU,   E2UPDF,   E2VARB,   E2VECT,   E2ZERO,-
ERINIT,   FLBGF2,   FLINI2,   FRINIT,   FRSOUR,   G2BPIN,-
G2CLPU,   G2DIVU,   G2FROZ,   G2HANG,   G2IBLC,   G2INIT,-
G2LCAT,   G2NODE,   G2PRNT,   G2RESO,   G2SMOT,   G2TIME,-
G3SMOT,   G4SMOT,   GETKY2,   H2FLOT,   H2INIT,   H2MIXT,-
H2SOLF,   H2SCRI,   H2TRIN,   H3INIT,   H3SCRN,   HSHEAR,-
LHINI2,   M2AREA,   NODIT2,   PSRED2,   PSREDU,   PSWRT2,-
PSWRTU,   PTIMP2,   ROGERC,   SETUPU,   SHORTG,   TIINI2,-
TIPRN2,   TVINIO,   TVINI1,   WRINI2,-
[PERVAIZ.ULT.OBJ]UL2LIB/LIB
```

D.3.3 Synopsis of variables

The file STAR.DOC defines most of the variables in the common blocks of the STAR code.

```
    ------------------------------------
    SYNOPSIS OF VARIABLES IN 2-D ROUTINES
    ------------------------------------
   ------------------------
   REAL NON-ARRAY VARIABLES
   ------------------------

ALPHA2   USER DEFINED MAXIMUM ALLOWABLE VALUE OF THRDA2        (GETKY2)
         CONSTANT USED FOR SPATIAL CELL DIVISION
AMCHPR   THE MACH NUMBER USED BY 'PRIMITIVE' ROUTINE
AMCHFL   REFERENCE MACH NUMBER                                 (GETKY2)
AMPLTV   AMPLITUDE OF TIME-VARYING MASS FLOW RATE
AMWTFL   REFERENCE MOLECULAR MASS
BEPSPR   THE ENERGY  USED BY 'PRIMITIVE' ROUTINE
BETAA2   MAXIMUM PERCENTAGE POINTS OF # OF CELLS THAT          (GETKY2)
         CAN BE SPATIALLY ADAPTED
BGF2JA   FLUX TERM F2 USED IN CONJUNCTION WITH FINDING ITS JACOBIANS
BGF4JA   FLUX TERM F4 USED IN CONJUNCTION WITH FINDING ITS JACOBIANS
BGG3JA   FLUX TERM G3 USED IN CONJUNCTION WITH FINDING ITS JACOBIANS
BGG4JA   FLUX TERM G4 USED IN CONJUNCTION WITH FINDING ITS JACOBIANS
CFLNTI   MINIMUM CFL NUMBER                                    (GETKY2)
CFLXTI   MAXIMUM CFL NUMBER                                    (GETKY2)
DISTFL   REFERENCE FLUID CHARACTERISTIC LENGTH                 (GETKY2)
DELTA2   SPECIFIED PERCENTAGE OF THRCA2 = THRDA2*DELTA2        (GETKY2)
DTCNTI   MINIMUM CONSTANT TIME STEP OVER ALL THE CELLS         (GETKY2)
         IF THIS IS LESS THAN OR EQUAL TO ZERO THEN
         TEMPORAL ADAPTATION IS USED
DTMNTI   MINIMUM TIME STEP OVER ALL THE CELLS
EPSOTI   EPSILON CORRECTION FOR ZERO VALUE OF TEMPORAL         (GETKY2)
         CRITERION
EPS1MN   MINIMUM ALLOWABLE VALUE OF EPS1TI                     (GETKY2)
```

EPS1MX	MAXIMUM ALLOWABLE VALUE OF EPS1TI	(GETKY2)
EPS1TI	EPSILON USED FOR TEMPORAL RESOLUTION	(GETKY2)
EPSLE2	EPSILON : MAGNITUDE OF CRITERIA FOR CONVERGENCE	(GETKY2)
ERORE2	TRANSPORTS THE GLOBAL ERROR FROM CONVERGENCE ROUTINE DEPENDING UPON THE TYPE OF CONVERGENCE	
ERRMAX	MAXIMUM ERROR ABOVE WHICH EPS1TI WILL BE DECREASED	(GETKY2)
ERRMTI	MAXIMUM ERROR ALLOWED BEFORE TIME-STEPS ARE REDUCED	(GETKY2)
ERRMIN	MINIMUM ERROR BELOW WHICH EPS1TI WILL BE INCREASED	(GETKY2)
FCTRTI	FACTOR MULTIPLYING THE TIME-STEPS IF ERROR EXCEEDS A USER DEFINED MAXIMUM VALUE (SEE ERRMTI)	(GETKY2)
FLOWTV	INITIAL MASS FLOW OR THE MASS FLOW AT THE END OF A PERIOD IN A TIME-VARYING BOUNDARY CONDITION	
FREQTV	FREQUENCY OF OSCILLATIONS IN A TIME-VARYING BOUNDARY CONDITION	
FMREFL	REFERENCE FLUID HEAT OF FORMATION	
GAMMA2	USER DEFINED MNIIMUM ALLOWABLE VALUE OF THRCA2 CONSTANT USED FOR SPATIAL CELL MERGER	(GETKY2)
GAMAFL	REFERENCE RATIO OF SPECIFIC HEATS	
GAMAPR	RATIO OF SPECIFIC HEATS USED BY PRIMITIVE ROUTINE	
GFACE2	A FACTOR INVOLVING GAMMA IN VISCOUS FLOWS	(GETKY2)
OMEGE2	TEMPERATURE EXPONENT FOR VISCOSITY	(GETKY2)
PHIEH2	EQUIVALENCE RATIO FOR HYDROGEN FUEL INJECTION	
PRESCH	REFERENCE CHEMISTRY PRESSURE	(GETKY2)
PRESFL	REFERENCE FLUID PRESSURE	(GETKY2)
PRESFR	DIMENSIONAL REFERENCE FLUID PRESSURE	
PRESPR	THE PRESSURE USED BY 'PRIMITIVE' ROUTINE	
RHORFL	REFERENCE FLUID DENSITY	(GETKY2)
RHORFR	DIMENSIONAL REFERENCE FLUID DENSITY	
RHORPR	THE DENSITY USED BY 'PRIMITIVE' ROUTINE	
RREYE2	RECIPROCAL OF REYNOLD'S NUMBER	(GETKY2)
RSCHE2	RECIPROCAL OF SCHMIDT'S NUMBER	(GETKY2)
SDELE2	COEFFICIENT DELTA USED IN THE COMPUTATION OF ARTIFICIAL VISCOSITY AT A NODE	
SMAXE2	MAXIMUM COEFFICIENT OF ARTIFICIAL VISCOSITY	(GETKY2)
SMINE2	MINIMUM COEFFICIENT OF ARTIFICIAL VISCOSITY	(GETKY2)
SONDPR	THE SOUND SPEED USED BY 'PRIMITIVE' ROUTINE	
TEMP1C	TEMPERATURE FOR DETERMINING EQUILIBRIUM RATES	(GETKY2)
TEMP2C	TEMPERATURE FOR DETERMINING EQUILIBRIUM RATES	(GETKY2)
TEMP3C	TEMPERATURE FOR DETERMINING EQUILIBRIUM RATES	(GETKY2)
TEMPPR	THE TEMPERATURE USED BY 'PRIMITIVE' ROUTINE	
THRCA2	COLLAPSE THRESHOLD LIMIT	
THRDA2	DIVIDE THRESHOLD LIMIT	
TIMNTI	STARTING TIME OF THE RUN	(GETKY2)
TIMXTI	MAXIMUM TIME OF THE RUN	(GETKY2)
TREFCH	REFERENCE CHEMISTRY TEMPERATURE	(GETKY2)
TREFFL	REFERENCE FLUID TEMPERATURE	(GETKY2)
TRIGCH	CHEMISTRY TRIGGER TEMPERATURE (FROZEN BELOW TRIGCH)	(GETKY2)
UCOMFR	DIMENSIONAL REFERENCE FLUID VELOCITY (U-COMP)	
UCOMPR	THE VELOCITY COMPONENT USED BY 'PRIMITIVE' ROUTINE	
VCOMFR	DIMENSIONAL REFERENCE FLUID VELOCITY (V-COMP)	
VCOMPR	THE VELOCITY COMPONENT USED BY 'PRIMITIVE' ROUTINE	
UGASFL	UNIVERSAL GAS CONSTANT	
UREFFL	REFERENCE FLUID VELOCITY	
WDREFL	REFERENCE FLUID SOURCE TERM	
YNRTCH	MASS FRACTIONS OF THE INERT SPECIES	

```
---------------------------
INTEGRAL NON-ARRAY VARIABLES
---------------------------

IADDH2   PARAMETER INDICATING IF SPECIAL FUEL INJECTION IS USED
IBASH2   THE BASE NODE IF FUEL IS TO ADDED AT A PLANE SURFACE
IDBGA2   DEBUG PARAMETER FOR ADAPTATION ROUTINES                (GETKY2)
IDBGCH   DEBUG PARAMETER FOR CHEMISTRY ROUTINES                 (GETKY2)
           -1 : WRITE EACH STEP
IDBGE2   DEBUG PARAMETER FOR EULER ROUTINES (E2 ROUTINES)       (GETKY2)
IDBGFR   DEBUG PARAMETER FOR REFERENCE ROUTINES (FR ROUTINES)   (GETKY2)
IDBGFL   DEBUG PARAMETER FOR FLUID ROUTINES (FL ROUTINES)       (GETKY2)
IDBGG2   DEBUG PARAMETER FOR GRID ROUTINES (G2 ROUTINES)        (GETKY2)
IDBGTI   DEBUG PARAMETER FOR TEMPORAL ROUTINES (TI ROUTINES)    (GETKY2)
IMGL     CURRENT SPATIAL LEVEL OF CELLS
IMPLTI   PARAMETER INDICATING USE OF IMPLICIT SOURCE TERMS      (GETKY2)
          IMPLTI: 1 FOR EXPLICIT; 0 FOR IMPLICIT
ITGL     CURRENT TEMPORAL LEVEL OF CELLS
JCARDS   CARD READER
JDEBUG   DEBUG UNIT FOR ALL DEBUG DUMPS
JDUMYN   DUMMY UNITS (N = 1,2,3,4)
JHISTO   HISTORY FILE -- STATISTICAL DATA FOR EACH ITERATION
JOUTAL   OUTPUT FILE  -- CONTAINS ALL THE OUTPUT
JPNTRE   CONTAINS ALL THE POINTER INFORMATION FOR RESTART PURPOSES
JPNTWR   WRITES ALL THE POINTER INFORMATION FOR RESTART PURPOSES
JPRINT   PRINT UNIT
JREADC   INPUTC.DAT   -- CONTAINS CHEMISTRY VARIABLES
JREADD   INPUTD.DAT   -- CONTAINS INITIAL DPENDENT VARIABLES
JREADF   INPUTF.DAT   -- CONTAINS OUTLET CONDITIONS
JREADG   INPUTG.DAT   -- CONTAINS GEOMETRIC INFORMATION
JREADI   INPUTI.DAT   -- CONTAINS INPUT RECORDS
JREADS   UNIT FOR READING THE SCHEDULE INPUT PROGRAM            (GETKY2)
         IF A SCHEDULE PROGRAM IS SUPPLIED SET JREADS .NE. 0
JTERMI   TERMINAL INPUT
JTERMO   TERMINAL OUTPUT
K1ADA2   FIRST  KEY VARIABLE FOR SPATIAL ADAPTATION             (GETKY2)
K2ADA2   SECOND KEY VARIABLE FOR SPATIAL ADAPTATION             (GETKY2)
KADPTI   KEY VARIABLE FOR TEMPORAL ADAPTATION                   (GETKY2)
KCHKA2   PARAMETER FOR CHECKING THE SUPERCELL AND NEIGHBOUR-    (GETKY2)
         CELL CALCULATIONS.  INPUT IN BINARY CODED VALUE
         1: CHECK AFTER G2DIVO (DIVIDE CELL)
         2: CHECK AFTER G2CLPO (MERGE CELLS)
         4: CHECK BEFORE COLLAPSING CELLS
         8: CHECK BEFORE DIVIDING CELLS
KDEBUG   OUTPUT (DEBUG) PARAMETER
KDIFTI   PARAMETER INDICATING THAT TIME-STEPS ARE TO BE         (GETKY2)
         REDUCED IF THERE EXIST LARGE DIFFERENCES IN SPECIES
         MASS FRACTION FOR THE SAME CELL
KDPENI   OPTION PARAMETER FOR SETTING DEPENDENT VARIABLES       (GETKY2)
         IN DPINIT
           1: READ FROM INPUT FILE -- AT ALL NODES
           2: SET UNIFORM VALUES
           3: SET LINEARLY VARYING VALUES FROM INLET TO OUTLET
KEQNE2   INDICATES THE EQUATION NUMBER TO BE USED FOR GENERATING
         CONVERGENCE HISTORY DATA
KFACTI   PARAMETER INDICATING THAT TIME-STEPS ARE TO BE         (GETKY2)
         USED IN CONJUNCTION WITH FCTRTI
```

```
KHAFEZ   OPTION PARAMETER FOR HAFEZ DOMINANT EIGENVALUE
KMERA2   PARAMETER INDICATING IF THE COLLAPSING OF CELLS IS     (GETKY2)
         TO BE DONE
KL0000-KUEFFF  HEXADECIMAL INTEGERS IN HEXCOD.INC
KONVE2   TYPE OF CONVERGENCE CRITERIA
           1: AVERAGE ERRORS ARE CHECKED
           2: MAXIMUM ERRORS ARE CHECKED
           3: RMS     ERRORS ARE CHECKED
KORDER   PARAMETER INDICATING IF THERE ARE                      (GETKY2)
         NON-ELEMENTARY REACTIONS
KPERFR   PARAMETER INDICATING IF PERIODIC BOUNDARY CONDITIONS   (GETKY2)
         ARE TO BE USED
KPLTA2   PARAMETER INDICATING IF SPATIAL THRESHOLD PLOTS ARE    (GETKY2)
         NEEDED
KROGER   PARAMETER INDICATING TYPE OF CHEMISTRY MODEL           (GETKY2)
           0: NO SPECIAL MODEL
           1: ROGER AND CHINITZ MODEL
           2: LIGHT HILL DISSOCIATION  MODEL
           3: FROZEN IDEAL GAS MODEL
KSRTE2   RESTART PARAMETER                                      (GETKY2)
             0 : START A NEW RUN WITH A STRUCTURED GRID
          1000 : START A NEW RUN WITH A BLOCK STRUCTURED GRID
             1 : RESTART FROM A PREVIOUS RUN AND READ FROM
                 FORMATTED FILE
          1001 : RESTART FROM A PREVIOUS RUN AND READ FROM
                 UNFORMATTED FILE
KTIMTI   PARAMETER INDICATING IF RESULTS AT VARIOUS TIME        (GETKY2)
         INTERVALS ARE NEEDED
MALVG2   MAXIMUM ALLOWABLE LEVEL FOR FINE CELLS                 (GETKY2)
MMAXTI   MAXIMUM ALLOWABLE TEMPORAL LEVEL FOR CELLS             (6)
MBNDG2   MAXIMUM NUMBER OF BOUNDARY POINTS                      (1000)
MCELG2   MAXIMUM NUMBER OF CELLS                                (20000)
MCYCFR   MAXIMUM NUMBER OF CYCLES FOR PERIODIC B.C.'S           (GETKY2)
MEQNFL   THE MAXIMUM NUMBER OF EQUATIONS TO BE SOLVED           (10)
METHA2   VARIATION METHOD FOR SPATIAL ADAPTATION                (GETKY2)
           1: NODE BASED VALUE
           2: CELL BASED VALUE
           3: NODE BASED FIRST GRADIENT
           4: CELL BASED FIRST GRADIENT
           5: CELL BASED SECOND GRADIENT
           6: CELL BASED, FOR MULTIPLE VARIABLES INVOLVING
              GENERALIZED NORMAL DISTRIBUTION ...
MITEPS   NUMBER OF ITERATIONS AFTER WHICH EPS1TI IS DECREASED   (GETKY2)
MITRA2   NUMBER OF ITERATIONS AFTER WHICH SPATIAL ADAPTATION    (GETKY2)
         IS DONE; ZERO MEANS NO SPATIAL ADAPTATION
MITRE2   MAXIMUM NUMBER OF ITERATIONS ALLOWED                   (GETKY2)
MITRPS   NUMBER OF TIMES AFTER THE POINTER SYSTEM IS SAVED      (GETKY2)
MLVLG2   MAXIMUM NUMBER OF LEVELS OF GRIDS                      (5)
MNODG2   MAXIMUM NUMBER OF NODES                                (16000)
MREACH   THE MAXIMUM NUMBER OF REACTIONS                        (20)
MSPECH   THE MAXIMUM NUMBER OF SPECIES (INCLUDING INERT ONES)   (6)
MTHRA2   THE NUMBER OF TIMES OF SPATIAL ADAPTATION CYCLES       (GETKY2)
         WHICH AFTER THE THRESHOLD LIMITS WILL BE COMPUTED
MTYPA2   INDICATES CELL/NODE BASED CALCULATION FOR METHA2
           0: CELL BASED CALCULATION
           1: NODE BASED CALCULATION
MUMDH2   MAXIMUM NUMBER INJECTION POINTS ON A SURFACE           (100)
```

MUMNTV	MAXIMUM NUMBER NODES ON A SURFACE WHERE TIME-VARYING BOUNDARY CONDITIONS ARE USED	(100)
NAPAKY	MAXIMUM NUMBER OF REAL KEYS IN GETKY2	(42)
NBNDG2	ACTUAL TOTAL NUMBER OF BOUNDARY NODES	
NCELA2	TOTAL NUMBER OF UNDIVIDED CELLS OR CELLS WITH CENTERS	
NCELG2	ACTUAL TOTAL NUMBER OF CELLS	
NCELH2	TOTAL NUMBER OF INJECTION CELLS	
NCRSG2	MAXIMUM ALLOWABLE LEVEL OF COARSE CELLS FOR MULTIPLE GRIDS IN STEADY STATE SOLUTIONS IF NON-ZERO FOR UNSTEADY FLOWS THEN THE SMALLEST TIME-STEPS ARE USED NEXT TO THE BOUNDARIES	(GETKY2)
NCYCFR	CURRENT NUMBER OF CYCLES FOR PERIODIC B.C.'S	(GETKY2)
NEQBAS	NUMBER OF BASIC CONSERVATION EQUATIONS (4 FOR 2-D)	
NEQNFL	ACTUAL NUMBER OF EQUATIONS TO BE SOLVED	
NEQSCH	ACTUAL NUMBER OF SPECIES EQUATIONS TO BE SOLVED	
NGIVTI	MAXIMUM GIVEN LEVEL OF TEMPORAL CELLS	(GETKY2)
NHNGA2	TOTAL NUMBER OF HANGING NODES (MIDDLE EDGE NODES OF THE FACES FOR CELLS WITHOUT CENTERS. THESE ARE THE MIDDLE NODES OF THE SPATIAL INTERFACES	
NINRCH	ACTUAL NUMBER OF INERT SPECIES	(GETKY2)
NIPAKY	MAXIMUM NUMBER OF INTEGER KEYS IN GETKY2	(42)
NITRE2	CURRENT NUMBER OF ITERATION FOR TWO DIMENSIONAL CODE	
NLVLG2	CURRENT MAXIMUM LEVEL OF FINE CELLS	
NMAXTI	MAXIMUM CALCULATED LEVEL OF TEMPORAL CELLS	
NMOVTI	PARAMETER INDICATING NUMBER OF CELLS TO BE MOVED AWAY FROM THE NODIT, SO THAT TEMPORAL INTERFACE COULD BE RELOCATED TO A PLACE WHERE THERE ARE LESS TEMPORAL GRADIENTS	(GETKY2)
NNODA2	ACTUAL TOTAL NUMBER OF NODES AFTER SUBTRACTING THE HANGING NODES (SEE NHNGA2)	
NNODG2	ACTUAL TOTAL NUMBER OF NODES	
NPLCA2	NUMBER OF PLACES FOR CELL/NODE BASED CALCULATIONS EITHER NNODG2 OR NCELA2	
NREACH	ACTUAL NUMBER OF REACTIONS IN THE SYSTEM	(GETKY2)
NSPECH	ACTUAL NUMBER OF SPECIES (INCLUDING INERT ONES)	(GETKY2)
NUMDH2	TOTAL NUMBER OF INJECTION NODES	
NUMNTV	CURRENT NUMBER NODES ON A SURFACE WHERE TIME-VARYING BOUNDARY CONDITIONS ARE USED	
NXTDA2	NUMBER OF CELLS TO BE EXTENDED FOR ADAPTIVE GRIDS OR THE NUMBER OF BUFFER LAYER FOR SPATIALLY RESOLVED REGION	(GETKY2)

REAL ARRAY VARIABLES

AMWTCH(S)	REFERENCE ATOMIC WEIGHT FOR SPECIES S	(INPUTC.DAT)
BIGWJA(J)	THE JTH SOURCE TERM FOR FINDING JACOBIANS	
BMIACH(IS,IR)	THE DIFFERENCE OF STIOCHIOMETRIC COEFFICIENTS FOR SPECIES IS IN REACTION IR (IBETCH-IALPCH)	
CELLTI(LC)	THE TIME STEP FOR CELL LC	
CHNGA2(PL)	THE CHANGE COMPUTED BY THE ADAPTATION ROUTINES AT PL	
CHNGE2(J,IN)	THE JTH CHANGE COMPUTED BY THE INTEGRATION ROUTINE AT NODE IN	
DPENFR(J)	JTH DIMENSIONAL REFERENCE DEPENDENT VARIABLE	
DPENG2(J,IN)	JTH DEPENDENT VARIABLE AT NODE IN	
DPENJA(J)	JTH DEPENDENT TERM FOR FINDING JACOBIANS	
DXEWM2(IC)	METRIC FOR CELL IC (EAST-WEST FOR X)	

```
DXNSM2(IC)    METRIC FOR CELL IC (NORTH-SOUTH FOR X)
DYEWM2(IC)    METRIC FOR CELL IC (EAST-WEST FOR Y)
DYNSM2(IC)    METRIC FOR CELL IC (NORTH-SOUTH FOR Y)
ENEBCH(S)     ENERGY TERM (E/R)       FOR REACTION R          (INPUTC.DAT)
              (BACKWARD)
ENEECH(S)     ENERGY TERM (E/R)       FOR REACTION R          (INPUTC.DAT)
              (EQUILIBRIUM)
ENEFCH(S)     ENERGY TERM (E/R)       FOR REACTION R          (INPUTC.DAT)
              (FORWARD)
EXPBCH(S)     EXPONENT OF TEMPERATURE FOR REACTION R          (INPUTC.DAT)
              (BACKWARD)
EXPECH(S)     EXPONENT OF TEMPERATURE FOR REACTION R          (INPUTC.DAT)
              (EQUILIBRIUM)
EXPFCH(S)     EXPONENT OF TEMPERATURE FOR REACTION R          (INPUTC.DAT)
              (FORWARD)
ENTRCH(S)     REFERENCE ENTROPY  FOR SPECIES S, KJ/KMOL/K     (INPUTC.DAT)
FMHTCH(S)     HEAT OF FORMATION  FOR SPECIES S IN KJ/KMOL     (INPUTC.DAT)
GEOMG2(1,IN)  X-COORDINATE AT NODE IN
GEOMG2(2,IN)  Y-COORDINATE AT NODE IN
PREBCH(S)     PRE-EXPONENTIAL FACTOR  FOR REACTION R          (INPUTC.DAT)
              (BACKWARD)
PREECH(S)     PRE-EXPONENTIAL FACTOR  FOR REACTION R          (INPUTC.DAT)
              (EQUILIBRIUM)
PREFCH(S)     PRE-EXPONENTIAL FACTOR  FOR REACTION R          (INPUTC.DAT)
              (FORWARD)
PRESG2(IN)    PRESSURE AT NODE IN
PERIM2(IC)    PERIMETER OF CELL VOLUME FOR CELL IC
RAMWCH(S)     RECIPROCAL OF ATOMIC WEIGHT FOR SPECIES S
RVOLM2(IC)    RECIPROCAL OF CELL VOLUME FOR CELL IC
SIGGE2(IN)    ARTIFICIAL VISCOSITY COEFFICIENT AT NODE IN
SPBSCH(S),    SECOND COEFFICIENT IN THE CONSTANT PRESSURE     (INPUTC.DAT)
              SPECIFIC HEAT FOR S, KJ/KMOL/K
SPCPCH(S)     FIRST  COEFFICIENT IN THE CONSTANT PRESSURE     (INPUTC.DAT)
              SPECIFIC HEAT FOR S, KJ/KMOL/K
SPCVCH(S)     CONSTANT  VOLUME  SPECIFIC HEAT FOR S,          (INPUTC.DAT)
              KJ/KMOL/K
TEMPG2(IN)    TEMPERATURE AT NODE IN
WORKA2(IN)    TEMPORARY WORK STORAGE FOR A2COMN
YMAXCH(IS)    MAXIMUM MASS FRACTION FOR SPECIES S
YSPECH(IS)    INITIAL (REFERENCE) MASS FRACTIONS FOR SPECIES S (INPUTC.DAT)
YSPEPR(IS)    MASS FRACTION FOR SPECIES S AT A CERTAIN NODE AS
              USED BY THE 'PRIMITIVE' SUBROUTINE
```

INTEGRAL NON-ARRAY VARIABLES

```
IALPCH(S,R)   REACTANT COEFFICIENT FOR SPECIES S IN REACTION R (INPUTC.DAT)
IALOCH(S,R)   ORDER OF REACTION  FOR SPECIES S IN REACTION R   (INPUTC.DAT)
IBETCH(S,R)   PRODUCT  COEFFICIENT FOR SPECIES S IN REACTION R (INPUTC.DAT)
IBTOCH(S,R)   ORDER OF REACTION  FOR SPECIES S IN REACTION R   (INPUTC.DAT)
IBNDG2(1,IB)  VALUE OF THE BOUNDARY NODE (WHICH IS A NODE ITSELF)
IBNDG2(2,IB)  FIRST  BASE CELL ADJACENT TO THE BOUNDARY NODE
IBNDG2(3,IB)  SECOND BASE CELL ADJACENT TO THE BOUNDARY NODE
IBNDG2(4,IB)  BOUNDARY EDGE
IBNDG2(5,IB)  TYPE OF BOUNDARY CONDITION USED FOR IBN
ICELA2(LC)    SETS THE POINTER ARRAY  WHICH HOLDS THE UNDIVIDED CELLS,
```

```
                OR CELLS WITHOUT CENTERS
ICELG2(1,LC)    CENTER       NODE OF CELL LC
ICELG2(2,LC)    SOUTH-WEST NODE OF CELL LC
ICELG2(3,LC)    SOUTH      NODE OF CELL LC
ICELG2(4,LC)    SOUTH-EAST NODE OF CELL LC
ICELG2(5,LC)    EAST       NODE OF CELL LC
ICELG2(6,LC)    NORTH-EAST NODE OF CELL LC
ICELG2(7,LC)    NORTH      NODE OF CELL LC
ICELG2(8,LC)    NORTH-WEST NODE OF CELL LC
ICELG2(9,LC)    WEST       NODE OF CELL LC
ICELG2(10,LC)   SUPERCELL OF CELL LC
ICELH2(IH)      THE CELL NUMBER FOR THE INJECTION POINT IH
ICELTI(LC)      POINTER FOR TEMPORAL CELL LC (ALL CELLS AT SAME LEVEL
                ARE CONTIGUOUSLY STORED -- SEE ILVLTI)
ILVLA2(1,LV)    FIRST CELL AT TEMPORAL LEVEL LV
ILVLA2(2,LV)    LAST  CELL AT TEMPORAL LEVEL LV
ILVLG2(1,LV)    FIRST CELL AT LEVEL LV
ILVLG2(2,LV)    LAST  CELL AT LEVEL LV
ILVLG2(3,LV)    NUMBER OF CELLS AT LEVEL LV
ILVLTI(1,LV)    FIRST CELL AT TEMPORAL LEVEL LV
ILVLTI(2,LV)    LAST  CELL AT TEMPORAL LEVEL LV
IPASKY(KY)      ARRAY PASSING THE INTEGER KEYWORD NUMBER KY IN GETKY2
ITABCH(S,R)     TABLE OF REACTION COEFFICIENT FOR SPECIES S IN REACTION R
KAUXG2(LC)      AUXILLIARY INFORMATION ABOUT CELL LC
MARAKY(KY)      ARRAY PASSING THE REAL KEYWORD NUMBER KY IN GETKY2
                IS SET, I.E., IT'S KEYWORD IS CHANGED IN CURRENT SIMULATION
MARIKY(KY)      ARRAY PASSING THE INTEGER KEYWORD NUMBER KY IN GETKY2
                IS SET, I.E., IT'S KEYWORD IS CHANGED IN CURRENT SIMULATION
NBCPG2(X,1)     FIRST  BOUNDARY NODE POINTER FOR A CORNER X
NBCPG2(X,2)     SECOND BOUNDARY NODE POINTER FOR A CORNER X
                X IS 1,2,3,4 FOR SW, SE, NE, NW CORNERS
NEIBG2(1,IN)    SOUTH-WEST CELL OF NODE IN
NEIBG2(2,IN)    SOUTH-EAST CELL OF NODE IN
NEIBG2(3,IN)    NORTH-EAST CELL OF NODE IN
NEIBG2(4,IN)    NORTH-WEST CELL OF NODE IN
NODEH2(IH)      THE NODE NUMBER CORRESPONDING TO AN INJECTION POINT IH
NODETV(IT)      THE NODE NUMBER CORRESPONDING TO A SURFACE POINT IT
                WHERE TEMPORALLY VARYING BOUNDARY CONDITIONS ARE APPLIED
NSRKCH(IR)      NUMBER OF SPECIES IN REACTION IR
MRKCA2(LI)      CONTAINS THE LIST OF CELLS TO BE COLLAPSED, DURING SPATIAL
                ADAPTATION MANIPULATIONS.  IT ALSO CONTAINS THE LIST OF
                NUMBER OF NODES MINUS THE NUMBER OF HANGING NODES
MRKDA2(LI)      CONTAINS THE LIST OF CELLS TO BE DIVIDED, DURING SPATIAL
                ADAPTATION MANIPULATIONS.  IT ALSO CONTAINS THE LIST OF
                NUMBER OF HANGING NODES


          ---------------
          OTHER VARIABLES
          ---------------

MTITLE      CHARACTER*80  TITLE FOR THE CURRENT RUN
```

D.3.4 Listing of STAR code

A2ADP0

```
      SUBROUTINE A2ADP0

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.INC     '
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] TICOMN.INC/LIST'
      DIMENSION MEMBER(4)
      LOGICAL   IWRITE

C*********************************************************************
C
C       THIS SUBROUTINE PERFORMS THE GRID REALIGNMENT NEEDED FOR
C       ADAPTIVE GRIDDING.  IT FINDS THE CELLS WHICH NEED TO BE DIVIDED
C       OR COLLAPSED.  FINALLY IT FINDS THE QUADRUPLES OF CELLS WHICH
C       PREVIOUSLY CONSTITUTED A SINGLE CELL SO THAT THEY CAN BE
C       COLLAPSED.
C
C
C       THIS ROUTINE SHOULD BE USED INSTEAD OF A2VOUU.FOR IF SOME
C       ERRORS ARE EXPECTED OR IF DEBUG PRINT IS DESIRED.
C
C*********************************************************************

C       INITIALIZE THE NUMBER OF CELLS TO BE COLLAPSED AND DIVIDED

      NCELLC = 0
      NCELLD = 0
C
C       LOOP THROUGH ALL THE CEWIC CELLS
C       THE CELL NEEDNOT BE DIVIDED OR COLLAPSED IF (KCENT .NE. 0)

      DO 10 JCELL = 1, NCELA2

C          FIND THE ACTUAL CELL ICELL

           ICELL  = ICELA2(JCELL)

C          DECIDE UPON CELL OR NODE BASED METHOD

           IF (MTYPA2 .NE. 0) THEN
              CHNGSW = CHNGA2(ICELG2(2,ICELL))
              CHNGSE = CHNGA2(ICELG2(4,ICELL))
              CHNGNE = CHNGA2(ICELG2(6,ICELL))
              CHNGNW = CHNGA2(ICELG2(8,ICELL))
              CHNGAV = 0.25*(CHNGSW + CHNGSE + CHNGNE + CHNGNW)
           ELSE
C             NOTE THAT THE CHANGE IS STORED IN JCELL (NOT ICELL)
              CHNGAV = CHNGA2(JCELL)
```

```
            ENDIF

C           CHECK IF CELL DIVISION IS REQUIRED, THE CELL IS TO BE
C           DIVIDED IF THE CELL CHANGE IS MORE THAN THRDA2;
C           MAKE A LIST OF SUCH CELLS

            IF(CHNGAV. GT. THRDA2) THEN
              NCELLD         = NCELLD + 1
              MRKDA2(NCELLD) = ICELL
            ENDIF

C           CHECK IF CELL COLLAPSING IS REQUIRED, THE CELL IS TO BE
C           COLLAPSED IF THE CELL CHANGE IS LESS THAN THRCA2;
C           THE CELL CAN NOT BE COLLAPED IF (LEVEL .LE. 0) OR WHEN
C           THE MERGE PARAMETER KMERA2 EQUALS ZERO
C           THE CELL IS ALSO NOT COLLAPSED IF IT WAS GENERATED LESS
C           THAN SIX TIME-STRIDE UNITS BEFORE
C           MAKE A LIST OF SUCH CELLS

            IF (KMERA2 .NE. 0) THEN
              IF (CHNGAV .LT. THRCA2) THEN
                IF (ICELG2(10,ICELL) .NE. 0) THEN
                   KX = KAUXG2(ICELL)
                   IF (IAND(KX,KLOOFO) .EQ. 0) THEN
                      NCELLC         = NCELLC + 1
                      MRKCA2(NCELLC) = ICELL
                   ENDIF
                ENDIF
              ENDIF
            ENDIF

10      CONTINUE
C
C       PRINT OUT PARAMETERS
C
        IWRITE  = IDBGA2 .EQ. 10 .OR. IDBGA2 .GT. 1000

        IF (IWRITE)  THEN
           WRITE(JDEBUG,1000)
           WRITE(JDEBUG,1100)
           WRITE(JDEBUG,1200)
           WRITE(JDEBUG,1300)
           WRITE(JDEBUG,1400) (MRKDA2(I), I = 1, NCELLD)
           WRITE(JDEBUG,1500)
           WRITE(JDEBUG,1400) (MRKCA2(I), I = 1, NCELLC)
        ENDIF
C
C       EXTEND THE CELLS TO BE DIVIDED, IF NEED BE
C
        WORKA2(1) = NCELLD
        WORKA2(2) = NCELLC

        CALL A2EXTD

C       RESET THE NUMBER OF CELLS TO BE DIVIDED OR COLLAPSED

        NCELLD = NINT(WORKA2(1))
```

```
      NCELLC = NINT(WORKA2(2))
C
C     PRINT OUT PARAMETERS
C
      IF (IWRITE)  THEN
         WRITE(JDEBUG,1600)
         WRITE(JDEBUG,1400) (MRKDA2(I), I = 1, NCELLD)
         WRITE(JDEBUG,1700)
         WRITE(JDEBUG,1400) (MRKCA2(I), I = 1, NCELLC)
      ENDIF
C
C     ------------------
C     MERGER CONFIRMATION
C     ------------------
C
C     FIND THE SET OF THE CELLS WHICH MAKE UP A CELL TO BE COLLAPSED
C     IF ONLY FEW OF THESE FOUR WANT TO BE COLLAPSED THEN NON CAN
C     BE COLLAPSED, I.E, WE MUST FIND FOUR SUBCELLS WITH THE SAME
C     SUPERCELL (OBVIOUSLY THE SUBCELLS WILL THEN BE AT THE SAME
C     LEVEL).  THE CELLS ARE ARRANGED AS QUADRUPLES IN CONTIGUOUS
C     AREAS OF MRKCA2 ARRAY.
      IFIRST = 1
20    NOELEM = 0
      LCELL = MRKCA2(IFIRST)

      DO 30 JCELL = IFIRST, NCELLC
         ICELL = MRKCA2(JCELL)
         IF ( ICELG2(10,ICELL) .EQ. ICELG2(10,LCELL) ) THEN
            NOELEM         = NOELEM + 1
            MEMBER(NOELEM) = JCELL
         ENDIF
         IF (NOELEM .EQ. 4) GOTO 50
30    CONTINUE
C
C     LESS THAN FOUR CELLS ARE FOUND; SO DESTROY THESE CELLS
C
      DO 40 IELEM = 1, NOELEM
         MRKCA2(MEMBER(IELEM)) = MRKCA2(NCELLC)
         NCELLC                = NCELLC - 1
40    CONTINUE

      IF (NCELLC .LE. IFIRST) THEN
         GOTO 70
      ELSE
         GOTO 20
      ENDIF
C
C     FOUR CELLS ARE FOUND; ARRANGE THEM IN CONTIGUOUS AREA
C
50    DO 60 IELEM = 0, 3
         MDUMMY                  = MRKCA2(IFIRST+IELEM)
         MRKCA2(IFIRST+IELEM)    = MRKCA2(MEMBER(IELEM+1))
         MRKCA2(MEMBER(IELEM+1)) = MDUMMY
60    CONTINUE
      IFIRST = IFIRST + 4
      IF (IFIRST .LT. NCELLC) GOTO 20
```

```
70      CONTINUE
C
C       READJUST THE CELLS TO BE COLLAPSED
C
        NCELLC = (NCELLC/4)*4
C
C       PRINT OUT PARAMETERS
C
        IF (IWRITE) THEN
          WRITE(JDEBUG,1800)
          WRITE(JDEBUG,1900)
          DO 80 ISET = 1, NCELLC, 4
             MEM1   = MRKCA2(ISET  )
             MEM2   = MRKCA2(ISET+1)
             MEM3   = MRKCA2(ISET+2)
             MEM4   = MRKCA2(ISET+3)
             ISUP1  = ICELG2(10,MEM1)
             ISUP2  = ICELG2(10,MEM2)
             ISUP3  = ICELG2(10,MEM3)
             ISUP4  = ICELG2(10,MEM4)
             WRITE(JDEBUG,2000) MEM1,MEM2,MEM3,MEM4,ISUP1,ISUP2,ISUP3,ISUP4
80        CONTINUE
        ENDIF
C
C       ----------
C       MARK NODES
C       ----------
C
C       SINCE THE GRID-DIVIDE AND GRID-COLLAPSE ROUTINES CHANGE THE
C       CELL ASSIGNMENT (AND NOT THE NODE ASSIGNMENTS) TRANSLATE
C       THE PREVIOUS INFORMATION (LISTS) IN TERMS OF SOUTHWEST NODES
C
        DO 90 JCELL = 1, NCELLD
          ICELL         = MRKDA2(JCELL)
          MRKDA2(JCELL) = ICELG2(2,ICELL)
90      CONTINUE

C       MARK THE NODES FOR THE CELLS TO BE COLLAPSED

        DO 100 ISET = 1, NCELLC, 4
           KSWM1          = ICELG2(2,MRKCA2(ISET  ))
           KSWM2          = ICELG2(2,MRKCA2(ISET+1))
           KSWM3          = ICELG2(2,MRKCA2(ISET+2))
           KSWM4          = ICELG2(2,MRKCA2(ISET+3))
           MRKCA2(ISET  ) = KSWM1
           MRKCA2(ISET+1) = KSWM2
           MRKCA2(ISET+2) = KSWM3
           MRKCA2(ISET+3) = KSWM4
100     CONTINUE
C
C       PRINT OUT PARAMETERS
C
        IF (IWRITE) THEN
          WRITE(JDEBUG,2100)
          WRITE(JDEBUG,1400) (MRKDA2(I), I = 1, NCELLD)
          WRITE(JDEBUG,2200)
          DO 110 ISET = 1, NCELLC, 4
```

```
            KSWM1  = MRKCA2(ISET  )
            KSWM2  = MRKCA2(ISET+1)
            KSWM3  = MRKCA2(ISET+2)
            KSWM4  = MRKCA2(ISET+3)
            MEM1   = NEIBG2(3,KSWM1)
            MEM2   = NEIBG2(3,KSWM2)
            MEM3   = NEIBG2(3,KSWM3)
            MEM4   = NEIBG2(3,KSWM4)
            ISUP1  = ICELG2(10,MEM1)
            ISUP2  = ICELG2(10,MEM2)
            ISUP3  = ICELG2(10,MEM3)
            ISUP4  = ICELG2(10,MEM4)
            WRITE(JDEBUG,2000) MEM1,MEM2,MEM3,MEM4,ISUP1,ISUP2,ISUP3,ISUP4
110       CONTINUE
        ENDIF

C       -------------
C       GRID DIVISION
C       -------------
C
C       CALL THE GRID DIVIDE ROUTINE FOR ALL THE PREVIOUSLY
C       COLLECTED CELLS.

        DO 120 JNODE = NCELLD, 1, -1
          KSW   = MRKDA2 (JNODE)
          JCELL = NEIBG2 (3,KSW)
          IWARN = 0
C         SEE IF PRINT OUT IS NEEDED IN THE CASE AN ERROR IS DETECTED
C         IN THE DEBUG CHECK ROUTINES, IN THE CASE OF NO ERROR THIS
C         PRINT OUT WILL BE DELETED
          IF (IAND(KCHKA2,KL0008) .NE. 0) THEN
             JPRINT = JDUMY3
             OPEN(UNIT=JPRINT, FILE='G2PRNT.DAT', STATUS='NEW')
             WRITE(JPRINT,2400)
             CALL G2PRNT(15)
          ENDIF
          WRITE(6,*) ' A2ADP0: CELL TO BE DIVIDED IS ',JCELL
          CALL    G2DIV0 (JCELL,IWARN)
          IF (IWARN .NE. 0) THEN
             WRITE(JTERM0,2250) IWARN, JCELL
             IF (IWARN .EQ. 10) GOTO 115
          ENDIF
C         SEE IF DEBUG CHECK IS NEEDED
          IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
            NERR = 0
            CALL CHKBN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
            CALL CHKNC2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
            CALL CHKNN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
            CALL CHKSP2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
          ENDIF
115       IF (IAND(KCHKA2,KL0008) .NE. 0) THEN
             CLOSE(UNIT=JPRINT, DISP='DELETE')
             JPRINT = 7
          ENDIF
120     CONTINUE
C
C       ------------
```

```
C       GRID COLLAPE
C       ------------
C
C       GRID COLLAPSE PROCESSING

        DO 130 ISET = 1, NCELLC, 4
           KSWM1 = MRKCA2(ISET  )
           KSWM2 = MRKCA2(ISET+1)
           KSWM3 = MRKCA2(ISET+2)
           KSWM4 = MRKCA2(ISET+3)
           MEM1  = NEIBG2(3,KSWM1)
           MEM2  = NEIBG2(3,KSWM2)
           MEM3  = NEIBG2(3,KSWM3)
           MEM4  = NEIBG2(3,KSWM4)
           ISUP1 = ICELG2(10,MEM1)
           ISUP2 = ICELG2(10,MEM2)
           ISUP3 = ICELG2(10,MEM3)
           ISUP4 = ICELG2(10,MEM4)
           IWARN = 0
           IF (ISUP1 .NE. ISUP2 .OR. ISUP1 .NE. ISUP3 .OR.
     1         ISUP1 .NE. ISUP4                          ) THEN
               ZER1 = ISUP2
               ZER2 = ISUP3
               CALL ERRORM (21,'A2ADPO','ISUP2 ',ZER1,'ISUP3 ',ZER2,
     1                      JPRINT,'SUPERCELLS DO NOT MATCH '      )
           ENDIF
C          SEE IF PRINT OUT IS NEEDED IN THE CASE AN ERROR IS DETECTED
C          IN THE DEBUG CHECK ROUTINES, IN THE CASE OF NO ERROR THIS
C          PRINT OUT WILL BE DELETED
           IF (IAND(KCHKA2,KL0004) .NE. 0) THEN
             JPRINT = JDUMY3
             OPEN(UNIT=JPRINT, FILE='G2PRNT.DAT', STATUS='NEW')
             WRITE(JPRINT,2300)
             CALL G2PRNT(15)
           ENDIF
           WRITE(6,*) ' A2ADPO: CELL TO BE COLLAPSED IS ',ISUP1,
     1                MEM1,MEM2,MEM3,MEM4
           CALL G2CLPO (MEM1, MEM2, MEM3, MEM4, ISUP1, IWARN)
C          SEE IF DEBUG CHECK IS NEEDED
           IF (IWARN .NE. 0) WRITE(JTERM0,2250) IWARN, ICELL
           IF (IAND(KCHKA2,KL0002) .NE. 0) THEN
             NERR = 0
             CALL CHKBN2 (ISUP1, MEM1, MEM2, MEM3, MEM4, NERR, 'AFTCLP')
             CALL CHKNN2 (ISUP1, MEM1, MEM2, MEM3, MEM4, NERR, 'AFTCLP')
             CALL CHKNC2 (ISUP1, MEM1, MEM2, MEM3, MEM4, NERR, 'AFTCLP')
             CALL CHKSP2 (ISUP1, MEM1, MEM2, MEM3, MEM4, NERR, 'AFTCLP')
           ENDIF
           IF (IAND(KCHKA2,KL0004) .NE. 0) THEN
             CLOSE(UNIT=JPRINT, DISP='DELETE')
             JPRINT = 7
             IF (IWARN .NE. 0) THEN
               WRITE(JPRINT,2350)
               WRITE(6,*) ' A2ADPO: LOOK AT FOR007.DAT OR PRINT OUTPUT UNIT'
               CALL G2PRNT(15)
             ENDIF
           ENDIF
 130     CONTINUE
```

```
C       ------------
C       DELETE NODES
C       ------------
C
C       SEE IF PRINT OUT IS NEEDED IN THE CASE AN ERROR IS DETECTED
C       IN THE DEBUG CHECK ROUTINES, IN THE CASE OF NO ERROR THIS
C       PRINT OUT WILL BE DELETED
        IF (KCHKA2 .EQ. 15) THEN
           JPRINT = JDUMY3
           OPEN(UNIT=JPRINT, FILE='G2PRNT.DAT', STATUS='NEW')
           WRITE(JPRINT,2400)
           CALL G2PRNT(15)
        ENDIF

C       DELETE ALL THE POINTERS CORRESPONDING TO DELETED NODES

        IF (NCELLC .GT. 0) CALL G2NODE

C       SEE IF DEBUG CHECK IS NEEDED
        IF (KCHKA2 .EQ. 15) THEN
          NERR = 0
          CALL CHKBN2 (0, 0, 0, 0, 0, NERR, 'AFTNOD')
          CALL CHKNC2 (0, 0, 0, 0, 0, NERR, 'AFTNOD')
          CALL CHKNN2 (0, 0, 0, 0, 0, NERR, 'AFTNOD')
          CALL CHKSP2 (0, 0, 0, 0, 0, NERR, 'AFTNOD')
          CLOSE(UNIT=JPRINT, DISP='DELETE')
          JPRINT = 7
        ENDIF

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM A2ADPO' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(/10X,'CELLS TO BE  DIVIDED  BEFORE EXTENSION')
1400    FORMAT(20I5)
1500    FORMAT(/10X,'CELLS TO BE COLLAPSED BEFORE EXTENSION')
1600    FORMAT(/10X,'CELLS TO BE  DIVIDED  AFTER  EXTENSION')
1700    FORMAT(/10X,'CELLS TO BE COLLAPSED AFTER  EXTENSION')
1800    FORMAT(/10X,'CELLS TO BE COLLAPSED AFTER MERGE CONFIRMATION'/)
1900    FORMAT(  7X,'CELL 1',4X,'CELL 2',4X,'CELL 3',4X,'CELL 4',
     1           4X,'SUPER1',4X,'SUPER2',4X,'SUPER3',4X,'SUPER4')
2000    FORMAT( 8(5X,I5) )
2100    FORMAT(5X,'SOUTHWEST NODES OF THE CELLS TO BE DIVIDED')
2200    FORMAT(5X,'CELLS IN TERMS OF SOUTHWEST NODES TO BE COLLAPSED')
2250    FORMAT(5X,'WARNING #',I3,2X,'ISSUED FOR CELL',I5)
2300    FORMAT(1X,'POINTER SYSTEM JUST BEFORE ERROR OCCURED IN G2CLPO')
2350    FORMAT(1X,'POINTER SYSTEM JUST AFTER  ERROR OCCURED IN G2CLPO')
2400    FORMAT(1X,'POINTER SYSTEM JUST BEFORE ERROR OCCURED IN G2DIVO')

        RETURN
        END
```

A2ADUU

```
      SUBROUTINE A2ADPO
C                A2ADUU

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'HEXCOD.INC'
      INCLUDE 'TICOMN.INC'
      DIMENSION MEMBER(4)

C*********************************************************************

C     THIS SUBROUTINE PERFORMS THE GRID REALIGNMENT NEEDED FOR
C     ADAPTIVE GRIDDING.  IT FINDS THE CELLS WHICH NEED TO BE DIVIDED
C     OR COLLAPSED.  FINALLY IT FINDS THE QUADRUPLES OF CELLS WHICH
C     PREVIOUSLY CONSTITUTED A SINGLE CELL SO THAT THEY CAN BE
C     COLLAPSED.

C*********************************************************************

C     INITIALIZE THE NUMBER OF CELLS TO BE COLLAPSED AND DIVIDED

      NCELLC = 0
      NCELLD = 0
      IWARN = 0
C
C     LOOP THROUGH ALL THE CEWIC CELLS
C     THE CELL NEEDNOT BE DIVIDED OR COLLAPSED IF (KCENT .NE. 0)

      DO 10 JCELL = 1, NCELA2

         ICELL  = ICELA2(JCELL)
C        NOTE THAT THE CHANGE IS STORED IN JCELL (NOT ICELL)
         CHNGAV = CHNGA2(JCELL)

C        CHECK IF CELL DIVISION IS REQUIRED, THE CELL IS TO BE
C        DIVIDED IF THE CELL CHANGE IS MORE THAN THRDA2;
C        MAKE A LIST OF SUCH CELLS

         IF(CHNGAV. GT. THRDA2) THEN
           NCELLD         = NCELLD + 1
           MRKDA2(NCELLD) = ICELL
         ENDIF

C        CHECK IF CELL COLLAPSING IS REQUIRED, THE CELL IS TO BE
C        COLLAPSED IF THE CELL CHANGE IS LESS THAN THRCA2;
C        THE CELL CAN NOT BE COLLAPED IF (LEVEL .LE. 0) OR WHEN
C        THE MERGE PARAMETER KMERA2 EQUALS ZERO
C        THE CELL IS ALSO NOT COLLAPSED IF IT WAS GENERATED LESS
C        THAN TWO TIME-STRIDE UNITS BEFORE
C        MAKE A LIST OF SUCH CELLS

         IF (KMERA2 .NE. 0) THEN
```

```
            IF (CHNGAV .LT. THRCA2) THEN
              IF (ICELG2(10,ICELL) .NE. 0) THEN
                 KX = KAUXG2(ICELL)
                 IF (IAND(KX,KLOOFO) .EQ. 0) THEN
                    NCELLC         = NCELLC + 1
                    MRKCA2(NCELLC) = ICELL
                 ENDIF
              ENDIF
            ENDIF
          ENDIF

10      CONTINUE
C
C       EXTEND THE CELLS TO BE DIVIDED, IF NEED BE
C
        WORKA2(1) = NCELLD
        WORKA2(2) = NCELLC
C
        CALL A2EXTD

C       RESET THE NUMBER OF CELLS TO BE DIVIDED OR COLLAPSED

        NCELLD = NINT(WORKA2(1))
        NCELLC = NINT(WORKA2(2))
C
C       -------------------
C       MERGER CONFIRMATION
C       -------------------
C
C       FIND THE SET OF THE CELLS WHICH MAKE UP A CELL TO BE COLLAPSED
C       IF ONLY FEW OF THESE FOUR WANT TO BE COLLAPSED THEN NONE CAN
C       BE COLLAPSED, I.E, WE MUST FIND FOUR SUBCELLS WITH THE SAME
C       SUPERCELL (OBVIOUSLY THE SUBCELLS WILL THEN BE AT THE SAME
C       LEVEL).  THE CELLS ARE ARRANGED AS QUADRUPLES IN CONTIGUOUS
C       AREAS OF MRKCA2 ARRAY.

        IFIRST = 1
20      NOELEM = 0
        LCELL = MRKCA2(IFIRST)

        DO 30 JCELL = IFIRST, NCELLC
           ICELL = MRKCA2(JCELL)
           IF ( ICELG2(10,ICELL) .EQ. ICELG2(10,LCELL) ) THEN
              NOELEM         = NOELEM + 1
              MEMBER(NOELEM) = JCELL
           ENDIF
           IF (NOELEM .EQ. 4) GOTO 50
30      CONTINUE
C
C       LESS THAN FOUR CELLS ARE FOUND; SO DESTROY THESE CELLS
C
        DO 40 IELEM = 1, NOELEM
           MRKCA2(MEMBER(IELEM)) = MRKCA2(NCELLC)
           NCELLC                = NCELLC - 1
40      CONTINUE

        IF (NCELLC .LE. IFIRST) THEN
```

```
          GOTO 70
        ELSE
          GOTO 20
        ENDIF
C
C       FOUR CELLS ARE FOUND; ARRANGE THEM IN CONTIGUOUS AREA
C
50      DO 60 IELEM = 0, 3
          MDUMMY                  = MRKCA2(IFIRST+IELEM)
          MRKCA2(IFIRST+IELEM)    = MRKCA2(MEMBER(IELEM+1))
          MRKCA2(MEMBER(IELEM+1)) = MDUMMY
60      CONTINUE
        IFIRST = IFIRST + 4
        IF (IFIRST .LT. NCELLC) GOTO 20

70      CONTINUE
C
C       READJUST THE CELLS TO BE COLLAPSED
C
        NCELLC = (NCELLC/4)*4
C
C       ----------
C       MARK NODES
C       ----------
C
C       SINCE THE GRID-DIVIDE AND GRID-COLLAPSE ROUTINES CHANGE THE
C       CELL ASSIGNMENT (AND NOT THE NODE ASSIGNMENTS) TRANSLATE
C       THE PREVIOUS INFORMATION (LISTS) IN TERMS OF SOUTHWEST NODES
C
        DO 90 JCELL = 1, NCELLD
          ICELL         = MRKDA2(JCELL)
          MRKDA2(JCELL) = ICELG2(2,ICELL)
90      CONTINUE

C       MARK THE NODES FOR THE CELLS TO BE COLLAPSED

        DO 100 ISET = 1, NCELLC, 4
          KSWM1          = ICELG2(2,MRKCA2(ISET  ))
          KSWM2          = ICELG2(2,MRKCA2(ISET+1))
          KSWM3          = ICELG2(2,MRKCA2(ISET+2))
          KSWM4          = ICELG2(2,MRKCA2(ISET+3))
          MRKCA2(ISET  ) = KSWM1
          MRKCA2(ISET+1) = KSWM2
          MRKCA2(ISET+2) = KSWM3
          MRKCA2(ISET+3) = KSWM4
100     CONTINUE

C       -------------
C       GRID DIVISION
C       -------------
C
C       CALL THE GRID DIVIDE ROUTINE FOR ALL THE PREVIOUSLY
C       COLLECTED CELLS.

        DO 120 JNODE = NCELLD, 1, -1
          KSW   = MRKDA2 (JNODE)
          JCELL = NEIBG2 (3,KSW)
```

```
          CALL    G2DIVO (JCELL,IWARN)
120     CONTINUE
C
C       ------------
C       GRID COLLAPE
C       ------------
C
C       GRID COLLAPSE PROCESSING

        DO 130 ISET = 1, NCELLC, 4
           KSWM1 = MRKCA2(ISET  )
           KSWM2 = MRKCA2(ISET+1)
           KSWM3 = MRKCA2(ISET+2)
           KSWM4 = MRKCA2(ISET+3)
           MEM1  = NEIBG2(3,KSWM1)
           MEM2  = NEIBG2(3,KSWM2)
           MEM3  = NEIBG2(3,KSWM3)
           MEM4  = NEIBG2(3,KSWM4)
           ISUP1 = ICELG2(10,MEM1)
C          ISUP2 = ICELG2(10,MEM2)
C          ISUP3 = ICELG2(10,MEM3)
C          ISUP4 = ICELG2(10,MEM4)
C          IF (ISUP1 .NE. ISUP2 .OR. ISUP1 .NE. ISUP3 .OR.
C     1               ISUP1 .NE. ISUP4                          ) THEN
C               ZER1 = ISUP2
C               ZER2 = ISUP3
C               CALL ERRORM (21,'A2ADPO','ISUP2 ',ZER1,'ISUP3 ',ZER2,
C     1                            JPRINT,'SUPERCELLS DO NOT MATCH '      )
C          ENDIF
           CALL G2CLPO (MEM1, MEM2, MEM3, MEM4, ISUP1, IWARN)
130     CONTINUE

        RETURN
        END
```

A2CEWC

```
        SUBROUTINE A2CEWC

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'HEXCOD.INC'
        INCLUDE 'IOCOMN.INC'

C***********************************************************************

C       THIS SUBROUTINE COMPUTES THE NUMBER NCELA2 OF "CEWIC" CELLS,
C       CEWIC IS THE ACRONYM FOR 'CELLS WITHOUT CENTER', I.E., THE
C       NON-MULTIPLE-GRID CELLS.  IT ALSO SETS THE POINTER ARRAY
C       ICELA2 WHICH HOLDS THE CEWIC CELLS.
```

```
C***********************************************************************

C       INITIALIZE THE NUMBER OF CEWIC CELLS

        NCELA2 = 0

C       INITIALIZE THE HISTORY DECREMENT FOR THE RECENTLY DIVIDED CELLS

        NINCHS = 16
C
C       LOOP THROUGH ALL THE CELLS ON ALL THE BASIC AND FINER LEVELS

CVD$    NODEPCHK
        DO 10 ICELL = ILVLG2(1,0), NCELG2

C          FIND THE CENTER NODE

           KCENT  = ICELG2(1,ICELL)

           IF (KCENT .EQ. 0) THEN
C               DECREASE THE TEMPORAL LEVEL BYTE IF NEED BE,
C               NOTE THAT IF THE CELL WERE DIVIDED MORE THAN ONCE
C               THEN THIS TREATMENT IS KEPT FROZEN
                KX             = KAUXG2(ICELL)
                IF (IAND(KX,KLOOFO) .GT. 0)
     1                KAUXG2(ICELL) = KAUXG2(ICELL) - NINCHS
c               IF (IAND(KX,KLFOOO) .EQ. 0) THEN
                   NCELA2         = NCELA2 + 1
                   ICELA2(NCELA2) = ICELL
c               ENDIF
           ENDIF

10      CONTINUE
C
C       PRINT OUT PARAMETERS
C

c       IF (IDBGA2 .NE. 13 .AND. IDBGA2 .LT. 1000)  RETURN
c       WRITE(JDEBUG,1000)
c       WRITE(JDEBUG,1100)
c       WRITE(JDEBUG,1200)
c       WRITE(JDEBUG,1300) NCELA2
c       WRITE(JDEBUG,1400) (ICELA2(I), I = 1, NCELA2)

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM A2CEWC' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(/10X,'NUMBER OF CEWIC CELLS = ',I5,/,
     1          10X,'THE CEWIC CELL POINTER IS :',/)
1400    FORMAT(20I5)

        RETURN
        END
```

A2EXTU

```
      SUBROUTINE A2EXTD
C                 A2EXTU

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'HEXCOD.INC'
      DIMENSION INB(8)

C***********************************************************************

C     THIS SUBROUTINE EXTENDS THE CLUSTER OF CELLS TO BE DIVIDED
C     BY A SPECIFIED NUMBER OF CELLS (NXTDA2) ON ALL THE SIDES
C     OF THE CELLS UNDER CONSIDERATION.  FOR EVERY CELL IN THE
C     CLUSTER, ALL ITS NEIGHBOUR CELLS ARE CHECKED, IF THESE
C     NEIGHBOUR CELLS ARE NOT IN THE CLUSTER, THEN THEY ARE
C     ADDED TO THE CLUSTER LIST.  THE NEIGBOUR CELLS ARE THEN
C     STORED IN A SEPERATE ARRAY; SUBSEQUENTLY (IF NXTDA2 > 1)
C     ONLY THE CELLS IN THIS ARRAY ARE CHECKED.

C***********************************************************************

C
C     SET THE NUMBER OF CELLS TO BE DIVIDED, COLLAPSED OR EXTENDED

      NCELLD = NINT(WORKA2(1))
      NCELLC = NINT(WORKA2(2))
      NCELDP = NCELLD + 1
      NEXTD  = 0
C
C     --------------------
C     CHECK DIVIDE CLUSTER
C     --------------------
C
      DO 60 JCELL = 1, NCELLD

         ICELL = MRKDA2(JCELL)
         KSW   = ICELG2( 2,ICELL)
         KS    = ICELG2( 3,ICELL)
         KSE   = ICELG2( 4,ICELL)
         KE    = ICELG2( 5,ICELL)
         KNE   = ICELG2( 6,ICELL)
         KN    = ICELG2( 7,ICELL)
         KNW   = ICELG2( 8,ICELL)
         KW    = ICELG2( 9,ICELL)

C        SET UP THE NEIGHBOUR CELLS OF THIS CELL
C
         INB(1) = NEIBG2(1,KSW)
```

```
      INB(2) = NEIBG2(2,KSE)
      INB(3) = NEIBG2(3,KNE)
      INB(4) = NEIBG2(4,KNW)

C     THE EXTENSION THROUGH A DIVIDED EDGE IS NOT NEEDED

      INB(5) = 0
      INB(6) = 0
      INB(7) = 0
      INB(8) = 0

      IF (KS .EQ. 0) INB(5) = NEIBG2(2,KSW)
      IF (KE .EQ. 0) INB(6) = NEIBG2(3,KSE)
      IF (KN .EQ. 0) INB(7) = NEIBG2(4,KNE)
      IF (KW .EQ. 0) INB(8) = NEIBG2(1,KNW)
C
C     IF THE LEVEL OF THE CORNER CELL IS HIGHER THAN THE
C     CLUSTER CELL'S LEVEL; THEN EXTENSION THROUGH THE
C     CORNER CELL IS NOT NEEDED; FIRST COMPUTE LEVEL (KLEVLC)
C     OF THE CLUSTER CELL AND THAT (KLEVLN) OF THE CORNER CELLS

      KLEVLC = ISHFT(IAND(KAUXG2(ICELL),KU000F),-16)
      IF (INB(1) .NE. 0) THEN
         KLEVLN = ISHFT(IAND(KAUXG2(INB(1)),KU000F),-16)
         IF (KLEVLN .GT. KLEVLC) INB(1) = 0
      ENDIF

      IF (INB(2) .NE. 0) THEN
         KLEVLN = ISHFT(IAND(KAUXG2(INB(2)),KU000F),-16)
         IF (KLEVLN .GT. KLEVLC) INB(2) = 0
      ENDIF

      IF (INB(3) .NE. 0) THEN
         KLEVLN = ISHFT(IAND(KAUXG2(INB(3)),KU000F),-16)
         IF (KLEVLN .GT. KLEVLC) INB(3) = 0
      ENDIF

      IF (INB(4) .NE. 0) THEN
         KLEVLN = ISHFT(IAND(KAUXG2(INB(4)),KU000F),-16)
         IF (KLEVLN .GT. KLEVLC) INB(4) = 0
      ENDIF


C     KELIG INDICATES THE NUMBER OF ELIGIBLE NEIGHBOUR CELLS
C     NOTE THAT THE ELIGIBLE CELLS WILL BE EVENTUALLY ALL
C     NON-ZERO NOW (ATMOST 8)

      KELIG = 0
      KELIG = KELIG + INB(1)
      KELIG = KELIG + INB(2)
      KELIG = KELIG + INB(3)
      KELIG = KELIG + INB(4)
      KELIG = KELIG + INB(5)
      KELIG = KELIG + INB(6)
      KELIG = KELIG + INB(7)
      KELIG = KELIG + INB(8)
```

```
            IF (KELIG .EQ. 0) GOTO 60
             -
C           NOW CHECK THE REST OF THE DIVIDE CLUSTER TO SEE IF THE
C           ELIGIBLE CELLS ARE INCLUDED THERE; IF SO THEY ARE NOT
C           THE ELIGIBLE CELLS.

            DO 40 KCELL = 1, NCELLD + NEXTD
              LCELL = MRKDA2(KCELL)
              IF (INB(1) .EQ. LCELL) INB(1) = 0
              IF (INB(2) .EQ. LCELL) INB(2) = 0
              IF (INB(3) .EQ. LCELL) INB(3) = 0
              IF (INB(4) .EQ. LCELL) INB(4) = 0
              IF (INB(5) .EQ. LCELL) INB(5) = 0
              IF (INB(6) .EQ. LCELL) INB(6) = 0
              IF (INB(7) .EQ. LCELL) INB(7) = 0
              IF (INB(8) .EQ. LCELL) INB(8) = 0
40          CONTINUE
C
            KELIG = 0
            KELIG = KELIG + INB(1)
            KELIG = KELIG + INB(2)
            KELIG = KELIG + INB(3)
            KELIG = KELIG + INB(4)
            KELIG = KELIG + INB(5)
            KELIG = KELIG + INB(6)
            KELIG = KELIG + INB(7)
            KELIG = KELIG + INB(8)
            IF (KELIG .EQ. 0) GOTO 60
C
C           NOW MARK THE CELLS WHICH ARE TO BE EXTENDED;  THE PAINTED
C           EDGES OR CORNERS (THROUGH WHICH EXTENSION OF THESE BOUNDARY
C           CELLS WILL NOT BE DONE), IS TEMPORARILY STORED IN WORKA2
C
            IF (INB(1) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(1)
               WORKA2(NPOINT) = 1
            ENDIF
            IF (INB(5) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(5)
               WORKA2(NPOINT) = 3
            ENDIF

            IF (INB(2) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(2)
               WORKA2(NPOINT) = 2
            ENDIF
            IF (INB(6) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(6)
               WORKA2(NPOINT) = 6
```

```
            ENDIF
            -
            IF (INB(3) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(3)
               WORKA2(NPOINT) = 4
            ENDIF
            IF (INB(7) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(7)
               WORKA2(NPOINT) = 12
            ENDIF

            IF (INB(4) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(4)
               WORKA2(NPOINT) = 8
            ENDIF
            IF (INB(8) .NE. 0) THEN
               NEXTD          = NEXTD + 1
               NPOINT         = NCELLD + NEXTD
               MRKDA2(NPOINT) = INB(8)
               WORKA2(NPOINT) = 9
            ENDIF

C           GO BACK FOR NEXT CLUSTER CELL

60       CONTINUE
C
C        ---------------
C        EXTEND BOUNDARY
C        ---------------
C
C        NOW EXTEND THE PREVIOUSLY EXTENDED CELLS; INDCEL INDICATES
C        THE EDGES OR CORNERS THROUGH WHICH EXTENSION HAD BEEN
C        PREVIOUSLY ACCOMPLISHED
C
         DO 130 INEXT = 1, NXTDA2-1

            JEXTD = 0

            DO 120 IEXTD = 1, NEXTD

               NPOINT = NCELLD + IEXTD
               ICELL  = MRKDA2(NPOINT)
               INDCEL = NINT(WORKA2(NPOINT))

C              SET UP NODE POINTERS FOR THIS CELL

               KSW    = ICELG2( 2,ICELL)
               KS     = ICELG2( 3,ICELL)
               KSE    = ICELG2( 4,ICELL)
               KE     = ICELG2( 5,ICELL)
               KNE    = ICELG2( 6,ICELL)
```

```
      KN     = ICELG2( 7,ICELL)
      KNW    = ICELG2( 8,ICELL)
      KW     = ICELG2( 9,ICELL)

C     SET UP THE NEIGHBOUR CELLS OF THIS CELL

      INB(1) = NEIBG2(1,KSW)
      INB(2) = NEIBG2(2,KSE)
      INB(3) = NEIBG2(3,KNE)
      INB(4) = NEIBG2(4,KNW)

      INB(5) = 0
      INB(6) = 0
      INB(7) = 0
      INB(8) = 0

C     THE EXTENSION THROUGH A DIVIDED EDGE IS NOT NEEDED

      IF (KS .EQ. 0) INB(5)   = NEIBG2(2,KSW)
      IF (KE .EQ. 0) INB(6)   = NEIBG2(3,KSE)
      IF (KN .EQ. 0) INB(7)   = NEIBG2(4,KNE)
      IF (KW .EQ. 0) INB(8)   = NEIBG2(1,KNW)
C
C     DON'T EXTEND THROUGH THE PARTICULAR EDGE OR CORNER
C     NORTHEAST
C
      IF (IAND(INDCEL,KL0001) .NE. 0) THEN
          INB(6) = 0
          INB(3) = 0
          INB(7) = 0
      ENDIF
C
C     NORTHWEST
C
      IF (IAND(INDCEL,KL0002) .NE. 0) THEN
          INB(7) = 0
          INB(4) = 0
          INB(8) = 0
      ENDIF
C
C     SOUTHWEST
C
      IF (IAND(INDCEL,KL0004) .NE. 0) THEN
          INB(8) = 0
          INB(1) = 0
          INB(5) = 0
      ENDIF
C
C     SOUTHEAST
C
      IF (IAND(INDCEL,KL0008) .NE. 0) THEN
          INB(5) = 0
          INB(2) = 0
          INB(6) = 0
      ENDIF
C
C     IF THE LEVEL OF THE CORNER CELL IS HIGHER . . .
```

```
C
      - KELIG = 0
        KELIG = KELIG + INB(1)
        KELIG = KELIG + INB(2)
        KELIG = KELIG + INB(3)
        KELIG = KELIG + INB(4)

        IF (KELIG .NE. 0) THEN
           KLEVLC = ISHFT(IAND(KAUXG2(ICELL),KUOOOF),-16)
           IF (INB(1) .NE. 0) THEN
              KLEVLN = ISHFT(IAND(KAUXG2(INB(1)),KUOOOF),-16)
              IF (KLEVLN .GT. KLEVLC) INB(1)  = 0
           ENDIF

           IF (INB(2) .NE. 0) THEN
              KLEVLN = ISHFT(IAND(KAUXG2(INB(2)),KUOOOF),-16)
              IF (KLEVLN .GT. KLEVLC) INB(2)  = 0
           ENDIF

           IF (INB(3) .NE. 0) THEN
              KLEVLN = ISHFT(IAND(KAUXG2(INB(3)),KUOOOF),-16)
              IF (KLEVLN .GT. KLEVLC) INB(3)  = 0
           ENDIF

           IF (INB(4) .NE. 0) THEN
              KLEVLN = ISHFT(IAND(KAUXG2(INB(4)),KUOOOF),-16)
              IF (KLEVLN .GT. KLEVLC) INB(4)  = 0
           ENDIF

        ENDIF

        KELIG = 0
        KELIG = KELIG + INB(1)
        KELIG = KELIG + INB(2)
        KELIG = KELIG + INB(3)
        KELIG = KELIG + INB(4)
        KELIG = KELIG + INB(5)
        KELIG = KELIG + INB(6)
        KELIG = KELIG + INB(7)
        KELIG = KELIG + INB(8)
        IF (KELIG .EQ. 0) GOTO 120
C
C       NOW CHECK THE REST OF THE DIVIDE CLUSTER
C
        DO 100 KCELL = 1, NCELLD + NEXTD + JEXTD
          LCELL = MRKDA2(KCELL)
          IF (INB(1) .EQ. LCELL) INB(1) = 0
          IF (INB(2) .EQ. LCELL) INB(2) = 0
          IF (INB(3) .EQ. LCELL) INB(3) = 0
          IF (INB(4) .EQ. LCELL) INB(4) = 0
          IF (INB(5) .EQ. LCELL) INB(5) = 0
          IF (INB(6) .EQ. LCELL) INB(6) = 0
          IF (INB(7) .EQ. LCELL) INB(7) = 0
          IF (INB(8) .EQ. LCELL) INB(8) = 0
100     CONTINUE
C
C       NOW PAINT THE CELLS WHICH ARE TO BE EXTENDED
```

```
C
      KELIG = 0
      KELIG = KELIG + INB(1)
      KELIG = KELIG + INB(2)
      KELIG = KELIG + INB(3)
      KELIG = KELIG + INB(4)
      KELIG = KELIG + INB(5)
      KELIG = KELIG + INB(6)
      KELIG = KELIG + INB(7)
      KELIG = KELIG + INB(8)
      IF (KELIG .EQ. 0) GOTO 120
C
      IF (INB(1) .NE. 0) THEN
          JEXTD          = JEXTD + 1
          NPOINT         = NCELLD + NEXTD + JEXTD
          MRKDA2(NPOINT) = INB(1)
          WORKA2(NPOINT) = 1
      ENDIF
      IF (INB(5) .NE. 0) THEN
          JEXTD          = JEXTD + 1
          NPOINT         = NCELLD + NEXTD + JEXTD
          MRKDA2(NPOINT) = INB(5)
          WORKA2(NPOINT) = 3
      ENDIF

      IF (INB(2) .NE. 0) THEN
          JEXTD          = JEXTD + 1
          NPOINT         = NCELLD + NEXTD + JEXTD
          MRKDA2(NPOINT) = INB(2)
          WORKA2(NPOINT) = 2
      ENDIF
      IF (INB(6) .NE. 0) THEN
          JEXTD          = JEXTD + 1
          NPOINT         = NCELLD + NEXTD + JEXTD
          MRKDA2(NPOINT) = INB(6)
          WORKA2(NPOINT) = 6
      ENDIF

      IF (INB(3) .NE. 0) THEN
          JEXTD          = JEXTD + 1
          NPOINT         = NCELLD + NEXTD + JEXTD
          MRKDA2(NPOINT) = INB(3)
          WORKA2(NPOINT) = 4
      ENDIF
      IF (INB(7) .NE. 0) THEN
          JEXTD          = JEXTD + 1
          NPOINT         = NCELLD + NEXTD + JEXTD
          MRKDA2(NPOINT) = INB(7)
          WORKA2(NPOINT) = 12
      ENDIF

      IF (INB(4) .NE. 0) THEN
          JEXTD          = JEXTD + 1
          NPOINT         = NCELLD + NEXTD + JEXTD
          MRKDA2(NPOINT) = INB(4)
          WORKA2(NPOINT) = 8
      ENDIF
```

```
            IF (INB(8) .NE. 0) THEN
                JEXTD          = JEXTD + 1
                NPOINT         = NCELLD + NEXTD + JEXTD
                MRKDA2(NPOINT) = INB(8)
                WORKA2(NPOINT) = 9
            ENDIF

C           GO BACK FOR NEXT MEMBER OF BOUNDARY

120       CONTINUE

C         ADJUST THE NUMBER OF CELLS TO BE DIVIDED AND EXTENDED

          NCELLD = NCELLD + NEXTD
          NEXTD  = JEXTD
C
C         GO BACK FOR NEXT LEVEL OF EXTENSION
C
130     CONTINUE
C
C       -------------------------
C       READJUST COLLAPSE CLUSTER
C       -------------------------
C
C       READJUST THE LIST OF CELLS TO BE MERGED; SOME OF THE CELLS
C       THAT ARE TO BE EXTENDED MAY BE ENLISTED HERE
C
        NCELLD = NCELLD + NEXTD

        DO 150 JCELL = NCELDP, NCELLD
          DO 140 KCELL = 1, NCELLC
            IF (MRKCA2(KCELL) .EQ. MRKDA2(JCELL)) THEN
              MRKCA2(KCELL) = MRKCA2(NCELLC)
              NCELLC        = NCELLC - 1
              GO TO 150
            ENDIF
140       CONTINUE
150     CONTINUE
C
C       RESET THE NUMBER OF CELLS TO BE DIVIDED OR COLLAPSED
C
        WORKA2(1) = NCELLD
        WORKA2(2) = NCELLC
C
C       ------------
C       NOMENCLATURE
C       -----------
C
C                 INB(4)       INB(7)       INB(3)
C                       +--------+---------+
C                       |8 KNW   7    KNE 6|
C                       |       KN         |
C                 INB(8)+9 KW          KE 5+INB(6)
C                       |       KS         |
C                       |2 KSW   3    KSE 4|
C                       +--------+---------+
C                 INB(1)       INB(5)       INB(2)
```

```
C
C
      RETURN
      END
```

A2EXTD

```
      SUBROUTINE A2EXTD

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.INC     '
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      DIMENSION INB(8)
      LOGICAL   IWRITE

C*********************************************************************
C
C       THIS SUBROUTINE EXTENDS THE CLUSTER OF CELLS TO BE DIVIDED
C       BY A SPECIFIED NUMBER OF CELLS (NXTDA2) ON ALL THE SIDES
C       OF THE CELLS UNDER CONSIDERATION.  FOR EVERY CELL IN THE
C       CLUSTER, ALL ITS NEIGHBOUR CELLS ARE CHECKED, IF THESE
C       NEIGHBOUR CELLS ARE NOT IN THE CLUSTER, THEN THEY ARE
C       ADDED TO THE CLUSTER LIST.  THE NEIGBOUR CELLS ARE THEN
C       STORED IN A SEPERATE ARRAY; SUBSEQUENTLY (IF NXTDA2 > 1)
C       ONLY THE CELLS IN THIS ARRAY ARE CHECKED.
C
C       THIS ROUTINE SHOULD BE USED INSTEAD OF A2EXTU.FOR IF SOME
C       ERRORS ARE EXPECTED OR IF DEBUG PRINT IS DESIRED.
C
C*********************************************************************

      IF (NXTDA2 .LT. 1) RETURN
C
C       SET THE NUMBER OF CELLS TO BE DIVIDED, COLLAPSED OR EXTENDED

      NCELLD = NINT(WORKA2(1))
      NCELLC = NINT(WORKA2(2))
      NCELDP = NCELLD + 1
      NEXTD  = 0

C       WANT DEBUG PRINT ?

      IWRITE = IDBGA2 .EQ. 11 .OR. IDBGA2 .GT. 1000
      NTIME  = 1
C
C       -------------------
C       CHECK DIVIDE CLUSTER
C       -------------------
C
```

```
      DO 60 JCELL = 1, NCELLD

C        FIND  THE ACTUAL CELL

         ICELL = MRKDA2(JCELL)

C        SET UP NODE POINTERS FOR THIS CELL

         KSW = ICELG2( 2,ICELL)
         KS  = ICELG2( 3,ICELL)
         KSE = ICELG2( 4,ICELL)
         KE  = ICELG2( 5,ICELL)
         KNE = ICELG2( 6,ICELL)
         KN  = ICELG2( 7,ICELL)
         KNW = ICELG2( 8,ICELL)
         KW  = ICELG2( 9,ICELL)

C        SET UP THE NEIGHBOUR CELLS OF THIS CELL
C
         INB(1) = NEIBG2(1,KSW)
         INB(2) = NEIBG2(2,KSE)
         INB(3) = NEIBG2(3,KNE)
         INB(4) = NEIBG2(4,KNW)

C        THE EXTENSION THROUGH A DIVIDED EDGE IS NOT NEEDED

         DO 10 IK = 1, 4
            INB(IK+4) = 0
10       CONTINUE

         IF (KS .EQ. 0) INB(5) = NEIBG2(2,KSW)
         IF (KE .EQ. 0) INB(6) = NEIBG2(3,KSE)
         IF (KN .EQ. 0) INB(7) = NEIBG2(4,KNE)
         IF (KW .EQ. 0) INB(8) = NEIBG2(1,KNW)
C
C        IF THE LEVEL OF THE CORNER CELL IS HIGHER THAN THE
C        CLUSTER CELL'S LEVEL; THEN EXTENSION THROUGH THE
C        CORNER CELL IS NOT NEEDED; FIRST COMPUTE LEVEL (KLEVLC)
C        OF THE CLUSTER CELL AND THAT (KLEVLN) OF THE CORNER CELLS

         KLEVLC = ISHFT(IAND(KAUXG2(ICELL),KU000F),-16)
         DO 20 IK = 1, 4
            IF (INB(IK) .NE. 0) THEN
               KLEVLN = ISHFT(IAND(KAUXG2(INB(IK)),KU000F),-16)
               IF (KLEVLN .GT. KLEVLC) INB(IK) = 0
            ENDIF
20       CONTINUE

C        KELIG INDICATES THE NUMBER OF ELIGIBLE NEIGHBOUR CELLS
C        NOTE THAT THE ELIGIBLE CELLS WILL BE EVENTUALLY ALL
C        NON-ZERO NOW (ATMOST 8)

         KELIG = 0
         DO 7001 I = 1, 8
            KELIG = KELIG + INB(I)
7001     CONTINUE
         IF (KELIG .EQ. 0) GOTO 60
```

```
C          NOW CHECK THE REST OF THE DIVIDE CLUSTER TO SEE IF THE
C          ELIGIBLE CELLS ARE INCLUDED THERE; IF SO THEY ARE NOT
C          THE ELIGIBLE CELLS.

           DO 40 KCELL = 1, NCELLD + NEXTD
             LCELL = MRKDA2(KCELL)
             DO 30 IK = 1, 8
                IF (INB(IK) .EQ. LCELL) INB(IK) = 0
30           CONTINUE
40         CONTINUE
C
           KELIG = 0
           DO 7002 I = 1, 8
              KELIG = KELIG + INB(I)
7002       CONTINUE
           IF (KELIG .EQ. 0) GOTO 60
C
C          NOW MARK THE CELLS WHICH ARE TO BE EXTENDED;  THE PAINTED
C          EDGES OR CORNERS (THROUGH WHICH EXTENSION OF THESE BOUNDARY
C          CELLS WILL NOT BE DONE), IS TEMPORARILY STORED IN WORKA2
C
           IPROD1 = 1
           IPROD2 = 3
           DO 50 IK = 1, 4
              IF (INB(IK) .NE. 0) THEN
                 NEXTD          = NEXTD + 1
                 NPOINT         = NCELLD + NEXTD
                 MRKDA2(NPOINT) = INB(IK)
                 WORKA2(NPOINT) = IPROD1
              ENDIF
              IPROD1 = IPROD1*2
              IF (INB(IK+4) .NE. 0) THEN
                 IF (IK .EQ. 4) IPROD2 = 9
                 NEXTD           = NEXTD + 1
                 NPOINT          = NCELLD + NEXTD
                 MRKDA2(NPOINT)  = INB(IK+4)
                 WORKA2(NPOINT)  = IPROD2
              ENDIF
              IPROD2 = IPROD2*2
50         CONTINUE

C          GO BACK FOR NEXT CLUSTER CELL

60       CONTINUE
C
C        PRINT OUT PARAMETERS
C
         IF (IWRITE) THEN
            WRITE(JDEBUG,1000)
            WRITE(JDEBUG,1100)
            WRITE(JDEBUG,1200)
            WRITE(JDEBUG,1300) NEXTD, 0
            WRITE(JDEBUG,1400) (MRKDA2(NCELLD+I), I = 1, NEXTD)
         ENDIF
C
C        ---------------
```

```
C       EXTEND BOUNDARY
C       ---------------
C
C       NOW EXTEND THE PREVIOUSLY EXTENDED CELLS; INDCEL INDICATES
C       THE EDGES OR CORNERS THROUGH WHICH EXTENSION HAD BEEN
C       PREVIOUSLY ACCOMPLISHED
C
        DO 130 INEXT = 1, NXTDA2-1

           JEXTD = 0

           DO 120 IEXTD = 1, NEXTD

C             FIND  THE ACTUAL CELL

              NPOINT = NCELLD + IEXTD
              ICELL  = MRKDA2(NPOINT)
              INDCEL = NINT(WORKA2(NPOINT))

C             SET UP NODE POINTERS FOR THIS CELL

              KSW    = ICELG2( 2,ICELL)
              KS     = ICELG2( 3,ICELL)
              KSE    = ICELG2( 4,ICELL)
              KE     = ICELG2( 5,ICELL)
              KNE    = ICELG2( 6,ICELL)
              KN     = ICELG2( 7,ICELL)
              KNW    = ICELG2( 8,ICELL)
              KW     = ICELG2( 9,ICELL)

C             SET UP THE NEIGHBOUR CELLS OF THIS CELL

              INB(1) = NEIBG2(1,KSW)
              INB(2) = NEIBG2(2,KSE)
              INB(3) = NEIBG2(3,KNE)
              INB(4) = NEIBG2(4,KNW)

              DO 70 IK = 1, 4
                 INB(IK+4) = 0
70            CONTINUE

C             THE EXTENSION THROUGH A DIVIDED EDGE IS NOT NEEDED

              IF (KS .EQ. 0) INB(5)   = NEIBG2(2,KSW)
              IF (KE .EQ. 0) INB(6)   = NEIBG2(3,KSE)
              IF (KN .EQ. 0) INB(7)   = NEIBG2(4,KNE)
              IF (KW .EQ. 0) INB(8)   = NEIBG2(1,KNW)
C
C             DON'T EXTEND THROUGH THE PARTICULAR EDGE OR CORNER
C             NORTHEAST
C
              IF (IAND(INDCEL,KL0001) .NE. 0) THEN
                  INB(6) = 0
                  INB(3) = 0
                  INB(7) = 0
              ENDIF
C
```

```
C             NORTHWEST
C
              IF (IAND(INDCEL,KL0002) .NE. 0) THEN
                  INB(7) = 0
                  INB(4) = 0
                  INB(8) = 0
              ENDIF
C
C             SOUTHWEST
C
              IF (IAND(INDCEL,KL0004) .NE. 0) THEN
                  INB(8) = 0
                  INB(1) = 0
                  INB(5) = 0
              ENDIF
C
C             SOUTHEAST
C
              IF (IAND(INDCEL,KL0008) .NE. 0) THEN
                  INB(5) = 0
                  INB(2) = 0
                  INB(6) = 0
              ENDIF
C
C             IF THE LEVEL OF THE CORNER CELL IS HIGHER . . .
C
              KELIG = 0
              DO 7003 I = 1, 4
                KELIG = KELIG + INB(I)
7003          CONTINUE

              IF (KELIG .NE. 0) THEN
                 KLEVLC = ISHFT(IAND(KAUXG2(ICELL),KU000F),-16)
                 DO 80 IK = 1, 4
                    IF (INB(IK) .NE. 0) THEN
                       KLEVLN = ISHFT(IAND(KAUXG2(INB(IK)),KU000F),-16)
                       IF (KLEVLN .GT. KLEVLC) INB(IK)  = 0
                    ENDIF
80                  CONTINUE
              ENDIF

              KELIG = 0
              DO 7004 I = 1, 8
                 KELIG = KELIG + INB(I)
7004          CONTINUE
              IF (KELIG .EQ. 0) GOTO 120
C
C             NOW CHECK THE REST OF THE DIVIDE CLUSTER
C
              DO 100 KCELL = 1, NCELLD + NEXTD + JEXTD
                LCELL = MRKDA2(KCELL)
                DO 90 IK = 1, 8
                   IF (INB(IK) .EQ. LCELL) INB(IK) = 0
90              CONTINUE
100           CONTINUE
C
C             NOW PAINT THE CELLS WHICH ARE TO BE EXTENDED
```

```
C
     -    KELIG = 0
          DO 7005 I = 1, 8
             KELIG = KELIG + INB(I)
7005         CONTINUE
          IF (KELIG .EQ. 0) GOTO 120
C
          IPROD1 = 1
          IPROD2 = 3
          DO 110 IK = 1, 4
             IF (INB(IK) .NE. 0) THEN
                JEXTD          = JEXTD + 1
                NPOINT         = NCELLD + NEXTD + JEXTD
                MRKDA2(NPOINT) = INB(IK)
                WORKA2(NPOINT) = IPROD1
             ENDIF
             IPROD1 = IPROD1*2
             IF (INB(IK+4) .NE. 0) THEN
                IF (IK .EQ. 4) IPROD2 = 9
                JEXTD          = JEXTD + 1
                NPOINT         = NCELLD + NEXTD + JEXTD
                MRKDA2(NPOINT) = INB(IK+4)
                WORKA2(NPOINT) = IPROD2
             ENDIF
             IPROD2 = IPROD2*2
110          CONTINUE

C            GO BACK FOR NEXT MEMBER OF BOUNDARY

120       CONTINUE

C         ADJUST THE NUMBER OF CELLS TO BE DIVIDED AND EXTENDED

          NCELLD = NCELLD + NEXTD
          NEXTD  = JEXTD
C
C         PRINT OUT PARAMETERS
C
          IF (IWRITE) THEN
              WRITE(JDEBUG,1300) NEXTD, INEXT
              WRITE(JDEBUG,1400) (MRKDA2(NCELLD+I), I = 1, NEXTD)
          ENDIF
C
C         GO BACK FOR NEXT LEVEL OF EXTENSION
C
130    CONTINUE
C
C      ------------------------
C      READJUST COLLAPSE CLUSTER
C      ------------------------
C
C      READJUST THE LIST OF CELLS TO BE MERGED; SOME OF THE CELLS
C      THAT ARE TO BE EXTENDED MAY BE ENLISTED HERE
C
       NCELLD = NCELLD + NEXTD

       DO 150 JCELL = NCELDP, NCELLD
```

```
          DO 140 KCELL = 1, NCELLC
            IF (MRKCA2(KCELL) .EQ. MRKDA2(JCELL)) THEN
              MRKCA2(KCELL) = MRKCA2(NCELLC)
              NCELLC        = NCELLC - 1
              GO TO 150
            ENDIF
140       CONTINUE
150     CONTINUE
C
C       RESET THE NUMBER OF CELLS TO BE DIVIDED OR COLLAPSED
C
        WORKA2(1) = NCELLD
        WORKA2(2) = NCELLC

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM A2EXTD' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'NUMBER OF EXTENDED CELLS',I5,2X,'AFTER PASS',I2/
     1         5X,'LIST OF EXTENDED CELLS IS :')
1400    FORMAT(20I5)

C
C       ------------
C       NOMENCLATURE
C       -----------
C
C                INB(4)       INB(7)       INB(3)
C                     +--------+---------+
C                     |8 KNW   7    KNE 6|
C                     |       KN         |
C                INB(8)+9 KW          KE 5+INB(6)
C                     |       KS         |
C                     |2 KSW   3    KSE 4|
C                     +--------+---------+
C                INB(1)       INB(5)       INB(2)
C
C


        RETURN
        END
```

A2GRDC

```
        SUBROUTINE A2GRDC

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] A2COMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
```

```
      INCLUDE '[.INC] IOCOMN.INC/LIST'

C*****************************************************************

C     THIS SUBROUTINE CALCULATES THE FIRST DIFFERENCE OF TWO CELL
C     QUANTITIES (DEPENDENT VARIABLES) FOR CEWIC CELLS.  THESE VARIABLES
C     ARE POINTED BY K1ADA2 AND K2ADA2.  THE NORMALIZED CELL  VALUES
C     ARE THEN STORED IN CHNGA2 AND WORKA2. SUBSEQUENTLY ONLY NORMALIZED
C     "CHI-SQUARE" VARIABLE IS USED FOR THE DECISION OF ADAPTATION.

C*****************************************************************
C
C     STEP THROUGH EACH CELL TO ACCUMULATE AVERAGE DIFFERENCE FOR
C     EACH SPATIAL ADAPTATION CRITERIA VARIABLE IN TWO DIRECTIONS

      AVGU1  = 0.
      AVGU2  = 0.

      DO 10 JCELL = 1, NPLCA2

C       POINT TO THE ACTUAL CELL

        ICELL = ICELA2(JCELL)

C       SET UP NODE POINTERS FOR THIS CELL

        KSW   = ICELG2(2,ICELL)
        KSE   = ICELG2(4,ICELL)
        KNE   = ICELG2(6,ICELL)
        KNW   = ICELG2(8,ICELL)

C       SAVE DEPENDENT VARIABLES AT ALL CELL CORNERS

        U1SW          = DPENG2(K1ADA2,KSW)
        U1SE          = DPENG2(K1ADA2,KSE)
        U1NE          = DPENG2(K1ADA2,KNE)
        U1NW          = DPENG2(K1ADA2,KNW)
        U1X           = U1NE +  U1SE - U1SW - U1NW
        U1Y           = U1NE +  U1NW - U1SW - U1SE

C       COMPUTE THE FIRST DIFFERENCE AT EACH CELL

        CHNGA2(JCELL) = U1X + U1Y
        AVGU1         = AVGU1 + CHNGA2(JCELL)

10    CONTINUE
C
      IF (K2ADA2 .NE. 0) THEN
        DO 20 JCELL = 1, NPLCA2
          ICELL         = ICELA2(JCELL)
          KSW           = ICELG2(2,ICELL)
          KSE           = ICELG2(4,ICELL)
          KNE           = ICELG2(6,ICELL)
          KNW           = ICELG2(8,ICELL)
          U2SW          = DPENG2(K2ADA2,KSW)
          U2SE          = DPENG2(K2ADA2,KSE)
          U2NE          = DPENG2(K2ADA2,KNE)
```

```
            U2NW          = DPENG2(K2ADA2,KNW)
            U2X           = U2NE +  U2SE - U2SW - U2NW
            U2Y           = U2NE +  U2NW - U2SW - U2SE
            WORKA2(JCELL) = U2X + U2Y
            AVGU2         = AVGU2 + WORKA2(JCELL)
20        CONTINUE
        ENDIF

C       COMPUTE THE AVERAGE CHANGE FOR ALL THE CELLS

        AVGU1 = AVGU1/NPLCA2
        AVGU2 = AVGU2/NPLCA2

C       COMPUTE THE VARIANCES OF THESE TWO QUANTITES

        VARU11 = 0.
        VARU12 = 0.
        VARU22 = 0.

        DO 30 JCELL = 1, NPLCA2
          CHNGA2(JCELL) = CHNGA2(JCELL) - AVGU1
          VARU11        = VARU11 + (CHNGA2(JCELL))**2
30      CONTINUE

        IF (K2ADA2 .NE. 0) THEN
          DO 40 JCELL = 1, NPLCA2
            WORKA2(JCELL) = WORKA2(JCELL) - AVGU2
            VARU12        = VARU12 + WORKA2(JCELL)*CHNGA2(JCELL)
            VARU22        = VARU22 + (WORKA2(JCELL))**2
40        CONTINUE
        ENDIF

        IF (NPLCA2 .EQ. 0 .OR. VARU11 .EQ. 0.) THEN
          ZER1 = VARU11
          ZER2 = NPLCA2
           CALL ERRORM (19,'A2GRDC','VARU11',ZER1,'NPLCA2',ZER2,
     1             JPRINT,'STANDARD DEVIATION ERROR')
        ENDIF

        VARU11 = VARU11/NPLCA2
        VARU12 = VARU12/NPLCA2
        VARU22 = VARU22/NPLCA2

C       COMPUTE THE DETERMINANT OF THE VARIANCE-COVARIANCE MATRIX

        DETERM = VARU11*VARU22 - VARU12*VARU12

C       COMPUTE THE INVERSE OF THE VARIANCE-COVARIANCE MATRIX

        IF (VARU22 .NE. 0.) THEN
           DETINV = 1./DETERM
           DUMMY  = VARU22
           VARU22 = VARU11*DETINV
           VARU11 = DUMMY *DETINV
           VARU12 =-VARU12*DETINV
        ELSE
           VARU11 = 1./VARU11
```

```
      ENDIF
C
C     WRITE THE RESULTS FOR SUBSEQUENT PLOTTING

      IF (KPLTA2 .NE. 0) THEN
         JPLOTA = 61
         WRITE (JPLOTA,1400) NCELA2,VARU11,VARU12,VARU22,AVGU1,AVGU2
         WRITE (JPLOTA,1500)(CHNGA2(NC), WORKA2(NC),NC=1,NPLCA2)
      ENDIF
C                                                                  T  -1
C     REASSIGN THE CHANGE VARIABLES AS AN ONE DIMENSIONAL ARRAY X  S  X

      DO 50 JCELL = 1, NPLCA2
        TERM1 = VARU11*CHNGA2(JCELL)**2
        TERM2 = VARU22*WORKA2(JCELL)**2
        TERM3 = VARU12*WORKA2(JCELL)*CHNGA2(JCELL)
        TERM4 = TERM1 + TERM2 + TERM3
        IF (TERM4 .LT. 0.) THEN
           ZER1 = JCELL
           ZER2 = TERM4
            CALL ERRORM (20,'A2GRDC','JCELL ',ZER1,'TERM4 ',ZER2,
     1          JPRINT,'COVARIANCE MATRIX IS NOT POSITIVE DEFINATE ?')
        ENDIF
        CHNGA2(JCELL) = SQRT(TERM4)
50    CONTINUE
C
C     PRINT OUT PARAMETERS
C
      IF (IDBGA2 .NE. 6 .AND. IDBGA2 .LT. 1000)  RETURN

      WRITE(JDEBUG,1000)
      WRITE(JDEBUG,1100)
      WRITE(JDEBUG,1200)
      WRITE(JDEBUG,1300) AVGU1, AVGU2, VARU11, VARU12, VARU22
      WRITE(JDEBUG,1500) (CHNGA2(I), I = 1, NPLCA2)

C     ----------------
C     FORMAT STATEMENTS
C     ----------------
C
1000  FORMAT(//10X,'----------------------' )
1100  FORMAT(  10X,'DEBUG PRINT FROM A2GRDC' )
1200  FORMAT(  10X,'----------------------'/)
1300  FORMAT(5X,'AVGU1  =',G14.5,5X,'AVGU2  =',G14.5/5X,
     1    'VARU11 =',G14.5,5X,'VARU12 =',G14.5,5X,'VARU22 =',G14.5,
     2    /10X,'CHANGES AFTER NORMALIZATION')
1400  FORMAT(I7,7G14.5)
1500  FORMAT(8G14.5)

      RETURN
      END
```

A2GRDN

```
      SUBROUTINE A2GRDN

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE CALCULATES THE FIRST DIFFERENCE OF TWO NODE
C     QUANTITIES (DEPENDENT VARIABLES) FOR CEWIC CELLS.  THESE VARIABLES
C     ARE POINTED BY K1ADA2 AND K2ADA2.  THE NORMALIZED CELL  VALUES
C     ARE THEN STORED IN CHNGA2 AND WORKA2. SUBSEQUENTLY ONLY NORMALIZED
C     "CHI-SQUARE" VARIABLE IS USED FOR THE DECISION OF ADAPTATION.


C*********************************************************************

C     ZERO OUT THE CHANGE AT EVERY PLACE

      DO 10 IPLAC = 1, NPLCA2
        CHNGA2(IPLAC) = 0.
        WORKA2(IPLAC) = 0.
10    CONTINUE
C
C     STEP THROUGH EACH CELL TO ACCUMULATE AVERAGE DIFFERENCE FOR
C     EACH SPATIAL ADAPTATION CRITERIA VARIABLE IN TWO DIRECTIONS

      DO 20 JCELL = 1, NCELA2

C       POINT TO THE ACTUAL CELL

        ICELL = ICELA2(JCELL)

C       SET UP NODE POINTERS FOR THIS CELL

        KSW = ICELG2(2,ICELL)
        KS  = ICELG2(3,ICELL)
        KSE = ICELG2(4,ICELL)
        KE  = ICELG2(5,ICELL)
        KNE = ICELG2(6,ICELL)
        KN  = ICELG2(7,ICELL)
        KNW = ICELG2(8,ICELL)
        KW  = ICELG2(9,ICELL)

C       SAVE DEPENDENT VARIABLES AT ALL CELL NODES, SINCE
C       VERY ACCURATE CALCULATION IS NOT DESIRED, WE ASSUME
C       THE EDGE VALUES AS THE AVERAGE VALUES

        U1SW = DPENG2(K1ADA2,KSW)
        U1SE = DPENG2(K1ADA2,KSE)
        U1NE = DPENG2(K1ADA2,KNE)
        U1NW = DPENG2(K1ADA2,KNW)
```

```
            U1S  = 0.50*(U1SW + U1SE)
            U1N  = 0.50*(U1NW + U1NE)
            U1E  = 0.50*(U1SE + U1NE)
            U1W  = 0.50*(U1SW + U1NW)
            U1C  = 0.25*(U1SW + U1SE + U1NE + U1NW)

C           COMPUTE CONTRIBUTION TO FIRST DIFFERENCE AT EACH CORNER NODE
C
C           SOUTHWEST CORNER
            CHNGA2(KSW) = CHNGA2(KSW) + U1S + U1W - 2.*U1SW
C
C           SOUTHEAST CORNER
            CHNGA2(KSE) = CHNGA2(KSE) + U1E - U1S
C
C           NORTHEAST CORNER
            CHNGA2(KNE) = CHNGA2(KNE) - U1N - U1E + 2.*U1SW
C
C           NORTHWEST CORNER
            CHNGA2(KNW) = CHNGA2(KNW) + U1N - U1W
C
C           ADD CONTRIBUTIONS IF SIDE NODES EXIST
C
            IF (KS .NE. 0) CHNGA2(KS) = CHNGA2(KS) +U1SE -U1SW +U1C -U1S
            IF (KE .NE. 0) CHNGA2(KE) = CHNGA2(KE) +U1NE -U1SE +U1E -U1C
            IF (KN .NE. 0) CHNGA2(KN) = CHNGA2(KN) +U1NE -U1NW +U1N -U1C
            IF (KW .NE. 0) CHNGA2(KW) = CHNGA2(KW) +U1NW -U1SW +U1C -U1W
C
20        CONTINUE
C
C         NOW CHECK IF THE SECOND CRITERIA VARIABLE EXISTS
C
          IF (K2ADA2 .NE. 0) THEN
            DO 30 JCELL = 1, NCELA2
              ICELL       = ICELA2(  JCELL)
              KSW         = ICELG2(2,ICELL)
              KS          = ICELG2(3,ICELL)
              KSE         = ICELG2(4,ICELL)
              KE          = ICELG2(5,ICELL)
              KNE         = ICELG2(6,ICELL)
              KN          = ICELG2(7,ICELL)
              KNW         = ICELG2(8,ICELL)
              KW          = ICELG2(9,ICELL)
              U2SW        = DPENG2(K2ADA2,KSW)
              U2SE        = DPENG2(K2ADA2,KSE)
              U2NE        = DPENG2(K2ADA2,KNE)
              U2NW        = DPENG2(K2ADA2,KNW)
              U2S         = 0.50*(U2SW + U2SE)
              U2N         = 0.50*(U2NW + U2NE)
              U2E         = 0.50*(U2SE + U2NE)
              U2W         = 0.50*(U2SW + U2NW)
              U2C         = 0.25*(U2SW + U2SE + U2NE + U2NW)
              WORKA2(KSW) = WORKA2(KSW) + U2S + U2W - 2.*U2SW
              WORKA2(KSE) = WORKA2(KSE) + U2E - U2S
              WORKA2(KNE) = WORKA2(KNE) - U2N - U2E + 2.*U2SW
              WORKA2(KNW) = WORKA2(KNW) + U2N - U2W
              IF (KS .NE. 0) WORKA2(KS) =WORKA2(KS) +U2SE -U2SW +U2C -U2S
              IF (KE .NE. 0) WORKA2(KE) =WORKA2(KE) +U2NE -U2SE +U2E -U2C
```

```
            IF (KN .NE. 0) WORKA2(KN) =WORKA2(KN) +U2NE -U2NW +U2N -U2C
            IF (KW .NE. 0) WORKA2(KW) =WORKA2(KW) +U2NW -U2SW +U2C -U2W
30          CONTINUE
         ENDIF

C        COMPUTE THE AVERAGE CHANGE FOR ALL THE NODES

         AVGU1  = 0.
         AVGU2  = 0.

         DO 40 IPLAC = 1, NPLCA2
           AVGU1 = AVGU1 + CHNGA2(IPLAC)
           AVGU2 = AVGU2 + WORKA2(IPLAC)
40       CONTINUE
C
         AVGU1 = AVGU1/NPLCA2
         AVGU2 = AVGU2/NPLCA2

C        COMPUTE THE VARIANCES OF THESE TWO QUANTITES

         VARU11 = 0.
         VARU12 = 0.
         VARU22 = 0.

         DO 50 INODE = 1, NPLCA2
           CHNGA2(INODE) = CHNGA2(INODE) - AVGU1
           VARU11        = VARU11 + (CHNGA2(INODE))**2
50       CONTINUE

         IF (K2ADA2 .NE. 0) THEN
           DO 60 INODE = 1, NPLCA2
             WORKA2(INODE) = WORKA2(INODE) - AVGU2
             VARU12        = VARU12 + WORKA2(INODE)*CHNGA2(INODE)
             VARU22        = VARU22 + (WORKA2(INODE))**2
60         CONTINUE
         ENDIF

         IF (NPLCA2 .EQ. 0 .OR. VARU11 .EQ. 0.) THEN
           ZER1 = VARU11
           ZER2 = NPLCA2
            CALL ERRORM (19,'A2GRDN','VARU11',ZER1,'NPLCA2',ZER2,
     1             JPRINT,'STANDARD DEVIATION ERROR')
         ENDIF
C
         VARU11 = VARU11/NPLCA2
         VARU12 = VARU12/NPLCA2
         VARU22 = VARU22/NPLCA2

C        COMPUTE THE DETERMINANT OF THE VARIANCE-COVARIANCE MATRIX

         DETERM = VARU11*VARU22 - VARU12*VARU12

C        COMPUTE THE INVERSE OF THE VARIANCE-COVARIANCE MATRIX

         IF (VARU22 .NE. 0.) THEN
            DETINV = 1./DETERM
            DUMMY  = VARU22
```

```
            VARU22 = VARU11*DETINV
            VARU11 = DUMMY *DETINV
            VARU12 =-VARU12*DETINV
         ELSE
            VARU11 = 1./VARU11
         ENDIF

C        WRITE THE RESULTS FOR SUBSEQUENT PLOTTING

         IF (KPLTA2 .NE. 0) THEN
            JPLOTA = 61
            WRITE (JPLOTA,1400) NCELA2,VARU11,VARU12,VARU22,AVGU1,AVGU2
            WRITE (JPLOTA,1500)(CHNGA2(NC), WORKA2(NC),NC=1,NPLCA2)
         ENDIF
C                                                                    T  -1
C        REASSIGN THE CHANGE VARIABLES AS AN ONE DIMENSIONAL ARRAY X  S  X

         DO 70 INODE = 1, NPLCA2
           TERM1 = VARU11*CHNGA2(INODE)**2
           TERM2 = VARU22*WORKA2(INODE)**2
           TERM3 = VARU12*WORKA2(INODE)*CHNGA2(INODE)
           TERM4 = TERM1 + TERM2 + TERM3
           IF (TERM4 .LT. 0.) THEN
              ZER1 = INODE
              ZER2 = TERM4
               CALL ERRORM (20,'A2GRDN','INODE ',ZER1,'TERM4 ',ZER2,
     1           JPRINT,'COVARIANCE MATRIX IS NOT POSITIVE DEFINATE ?')
           ENDIF
           CHNGA2(INODE) = SQRT(TERM4)
70       CONTINUE
C
C        PRINT OUT PARAMETERS
C
         IF (IDBGA2 .NE. 5 .AND. IDBGA2 .LT. 1000)  RETURN

         WRITE(JDEBUG,1000)
         WRITE(JDEBUG,1100)
         WRITE(JDEBUG,1200)
         WRITE(JDEBUG,1300) AVGU1, AVGU2, VARU11, VARU12, VARU22
         WRITE(JDEBUG,1500) (CHNGA2(I), I = 1, NPLCA2)

C        -----------------
C        FORMAT STATEMENTS
C        -----------------

1000     FORMAT(//10X,'----------------------' )
1100     FORMAT(  10X,'DEBUG PRINT FROM A2GRDN' )
1200     FORMAT(  10X,'----------------------'/)
1300     FORMAT(5X,'AVGU1  =',G14.5,5X,'AVGU2  =',G14.5/5X,
     1      'VARU11 =',G14.5,5X,'VARU12 =',G14.5,5X,'VARU22 =',G14.5,
     2      /10X,'CHANGES AFTER NORMALIZATION')
1400     FORMAT(I7,7G14.5)
1500     FORMAT(8G14.5)

         RETURN
         END
```

A2INIT

```
      SUBROUTINE A2INIT

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'HEXCOD.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'

      CHARACTER*15  METHOD(6)
      DATA METHOD / 'NODE VALUES    ',  'CELL VALUES    ',
     1              'NODE GRADIENTS ',  'CELL GRADIENTS ',
     2              'NODE LAPLACIANS',  'MAX CELL DIFF  '/

C*********************************************************************
C
C     THIS SUBROUTINE INITIALIZES THE CONSTANTS FOR THE ARRAYS USED
C     IN THE ADAPTIVE GRID ROUTINES.
C
C*********************************************************************
C
C     INITIALIZE THE VARIABLES USED FOR SPATIAL ADAPTATION

      ALPHA2 = APASKY(15)
      BETAA2 = APASKY(16)
      GAMMA2 = APASKY(17)
      DELTA2 = APASKY(18)
      THRDA2 = ALPHA2
      THRCA2 = GAMMA2*THRDA2
      THRCA2 = MIN(THRCA2,DELTA2)

C     SPATIAL ADAPTATION CRITERION VARIABLE
      K1ADA2 = IPASKY(11)
      K2ADA2 = IPASKY(12)

C     METHOD OF SPATIAL ADAPTATION
      METHA2 = IPASKY( 8)

C     NUMBER OF ADAPTATION CYCLES AFTER WHICH THRESHOLD LIMITS WILL
C     BE CHECKED
      MTHRA2 = IPASKY(16)

C     PARAMETER INDICATING IF THRESHOLD PLOTS ARE NEEDED
      KPLTA2 = IPASKY(20)

C     NUMBER OF CELLS TO BE EXTENDED
      NXTDA2 = IPASKY(22)

C     NUMBER OF ITERATIONS BETWEEN SPATIAL ADAPTATION OPERATIONS
      MITRA2 = IPASKY(26)
```

```
C       DEBUG UNIT
        IDBGA2 = IPASKY(15)

C30     PARAMETER INDICATING IF THE COLLAPSING OF CELLS IS TO BE DONE
        KMERA2 = IPASKY(30)

C       DEBUG CHECK CALCULATION PARAMETER FOR CHKNN2, CHKBN2 AND CHKSP2
        KCHKA2 = IPASKY(31)

C       CHECK ERRORS FOR METHA2 HERE



C       NODE OR CELL BASED CALCULATIONS
        MTYPA2 = IAND(METHA2,KL0001)

C       INITIALIZE THE POINTERS FOR THE CEWIC CELLS

        CALL A2CEWC
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGA2 .NE. 1 .AND. IDBGA2 .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) ALPHA2, BETAA2, GAMMA2, DELTA2, THRDA2,
     1                     THRCA2
        WRITE(JDEBUG,1400) K1ADA2, K2ADA2, NXTDA2, MTHRA2, KPLTA2,
     1                     MITRA2, IDBGA2, KMERA2, KCHKA2, MTYPA2
        WRITE(JDEBUG,1500) METHA2, METHOD(METHA2)

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM A2INIT' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X, 'ALPHA2 =', G14.5, 5X, 'BETAA2 =', G14.5,
     1        5X, 'GAMMA2 =', G14.5, 5X, 'DELTA2 =', G14.5/
     2        5X, 'THRDA2 =', G14.5, 5X, 'THRCA2 =', G14.5)
1400    FORMAT( 5X, 'SPATIAL ADAPTATION CRITERION   =', I5,
     1        10X, 'SECOND VARIABLE                 =', I5/
     2         5X, 'NUMBER OF CELLS TO BE EXTENDED =', I5,
     3        10X, 'MTHRA2 : # ADAPTATION CYCLES   =', I5/
     4         5X, 'KPLTA2 : THRESHOLD PLOTS ?     =', I5,
     5        10X, 'MITRA2 : ITERS B/W ADAPTATIONS =', I5/
     6         5X, 'DEBUG UNIT                     =', I5,
     7        10X, 'KMERA2 : COLLAPSE CELLS ?      =', I5/
     8         5X, 'KCHKA2 : CHKNN2 AND CHKSP2 ?   =', I5,
     9        10X, 'MTYPA2 : NODE OR CELL BASED    =', I5)
1500    FORMAT( 5X, 'METHOD OF ADPTATION            =', I5, 10X, A15)

        RETURN
        END
```

A2MDFU

```
      SUBROUTINE A2MDIF
C                A2MDFU

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'G2COMN.INC'

C**********************************************************************

C     THIS SUBROUTINE CALCULATES THE MAXIMUM FIRST DIFFERENCE OF TWO
C     CELL QUANTITIES (DEPENDENT VARIABLES).  THESE VARIABLES ARE
C     POINTED BY K1ADA2 AND K2ADA2.  THE NORMALIZED CELLS VALUES ARE
C     THEN STORED IN CHNGA2 AND WORKA2. SUBSEQUENTLY ONLY NORMALIZED
C     "CHI-SQUARE" VARIABLE IS USED FOR THE DECISION OF ADAPTATION.

C**********************************************************************

C     STEP THROUGH EACH CELL TO ACCUMULATE MAXIMUM AVERAGE DIFFERENCE
C     FOR EACH SPATIAL ADAPTATION CRITERIA VARIABLE

      AVGU1  = 0.
      AVGU2  = 0.

      DO 10 JCELL = 1, NPLCA2

        ICELL = ICELA2(JCELL)
        KSW   = ICELG2(2,ICELL)
        KSE   = ICELG2(4,ICELL)
        KNE   = ICELG2(6,ICELL)
        KNW   = ICELG2(8,ICELL)

C       SAVE DEPENDENT VARIABLES AT ALL CELL CORNERS

        U1SW          = DPENG2(K1ADA2,KSW)
        U1SE          = DPENG2(K1ADA2,KSE)
        U1NE          = DPENG2(K1ADA2,KNE)
        U1NW          = DPENG2(K1ADA2,KNW)
        U1AV          = 0.25*(U1SW + U1SE + U1NE + U1NW)
        U1SW          = U1SW - U1AV
        U1SE          = U1SE - U1AV
        U1NE          = U1NE - U1AV
        U1NW          = U1NW - U1AV
C       COMPUTE MAXIMUM (OR MINIMUM) FIRST DIFFERENCE FOR THE CELL
        U1MAX         = MAX (U1SW, U1SE, U1NE, U1NW)
        U1MIN         = MIN (U1SW, U1SE, U1NE, U1NW)
        IF (U1MAX .LT. ABS(U1MIN)) U1MAX = U1MIN
        CHNGA2(JCELL) = U1MAX
        AVGU1         = AVGU1 + U1MAX
```

```
10      CONTINUE
C           -
        K3ADA2 = K2ADA2

        IF (K3ADA2 .NE. 0) THEN
          IF (K2ADA2 .GT. 100) THEN
             K3ADA2 = K2ADA2 - 100
             DO 20 JCELL = 1, NPLCA2
               ICELL          = ICELA2(JCELL)
               KSW            = ICELG2(2,ICELL)
               KSE            = ICELG2(4,ICELL)
               KNE            = ICELG2(6,ICELL)
               KNW            = ICELG2(8,ICELL)
               U2SW           = DPENG2(K3ADA2,KSW)/DPENG2(1,KSW)
               U2SE           = DPENG2(K3ADA2,KSE)/DPENG2(1,KSE)
               U2NE           = DPENG2(K3ADA2,KNE)/DPENG2(1,KNE)
               U2NW           = DPENG2(K3ADA2,KNW)/DPENG2(1,KNW)
               U2AV           = 0.25*(U2SW + U2SE + U2NE + U2NW)
               U2SW           = U2SW - U2AV
               U2SE           = U2SE - U2AV
               U2NE           = U2NE - U2AV
               U2NW           = U2NW - U2AV
               U2MAX          = MAX (U2SW, U2SE, U2NE, U2NW)
               U2MIN          = MIN (U2SW, U2SE, U2NE, U2NW)
               IF (U2MAX .LT. ABS(U2MIN)) U2MAX = U2MIN
               WORKA2(JCELL) = U2MAX
               AVGU2          = AVGU2 + U2MAX
20           CONTINUE
          ELSE
             DO 25 JCELL = 1, NPLCA2
               ICELL          = ICELA2(JCELL)
               KSW            = ICELG2(2,ICELL)
               KSE            = ICELG2(4,ICELL)
               KNE            = ICELG2(6,ICELL)
               KNW            = ICELG2(8,ICELL)
               U2SW           = DPENG2(K3ADA2,KSW)
               U2SE           = DPENG2(K3ADA2,KSE)
               U2NE           = DPENG2(K3ADA2,KNE)
               U2NW           = DPENG2(K3ADA2,KNW)
               U2AV           = 0.25*(U2SW + U2SE + U2NE + U2NW)
               U2SW           = U2SW - U2AV
               U2SE           = U2SE - U2AV
               U2NE           = U2NE - U2AV
               U2NW           = U2NW - U2AV
               U2MAX          = MAX (U2SW, U2SE, U2NE, U2NW)
               U2MIN          = MIN (U2SW, U2SE, U2NE, U2NW)
               IF (U2MAX .LT. ABS(U2MIN)) U2MAX = U2MIN
               WORKA2(JCELL) = U2MAX
               AVGU2          = AVGU2 + U2MAX
25           CONTINUE
          ENDIF
C         ENDIF      (K2ADA2 .GT. 100)
        ENDIF
C       ENDIF       (K3ADA2 .NE. 0)

C       COMPUTE THE AVERAGE CHANGE FOR ALL THE CELLS
```

```
      AVGU1 = AVGU1/NPLCA2
      AVGU2 = AVGU2/NPLCA2

C     COMPUTE THE VARIANCES OF THESE TWO QUANTITES

      VARU11 = 0.
      VARU12 = 0.
      VARU22 = 0.

      DO 30 JCELL = 1, NPLCA2
        CHNGA2(JCELL) = CHNGA2(JCELL) - AVGU1
        VARU11        = VARU11 + (CHNGA2(JCELL))**2
30    CONTINUE

      IF (K3ADA2 .NE. 0) THEN
        DO 40 JCELL = 1, NPLCA2
          WORKA2(JCELL) = WORKA2(JCELL) - AVGU2
          VARU12        = VARU12 + WORKA2(JCELL)*CHNGA2(JCELL)
          VARU22        = VARU22 + (WORKA2(JCELL))**2
40      CONTINUE
      ENDIF
C
      VARU11 = VARU11/NPLCA2
      VARU12 = VARU12/NPLCA2
      VARU22 = VARU22/NPLCA2
      SD1    = SQRT(VARU11)
      SD2    = SQRT(VARU22)

C     COMPUTE THE DETERMINANT OF THE VARIANCE-COVARIANCE MATRIX

      DETERM = VARU11*VARU22 - VARU12*VARU12

C     COMPUTE THE INVERSE OF THE VARIANCE-COVARIANCE MATRIX

      IF (VARU22 .NE. 0.) THEN
         DETINV = 1./DETERM
         DUMMY  = VARU22
         VARU22 = VARU11*DETINV
         VARU11 = DUMMY *DETINV
         VARU12 =-VARU12*DETINV
      ELSE
         VARU11 = 1./VARU11
      ENDIF

C                                                              T  -1
C     REASSIGN THE CHANGE VARIABLES AS AN ONE DIMENSIONAL ARRAY X  S  X

      DO 50 JCELL = 1, NPLCA2
        TERM1 = VARU11*CHNGA2(JCELL)**2
        TERM2 = VARU22*WORKA2(JCELL)**2
        TERM3 = VARU12*WORKA2(JCELL)*CHNGA2(JCELL)
        TERM4 = TERM1 + TERM2 + TERM3
        CHNGA2(JCELL) = SQRT(TERM4)
50    CONTINUE

      RETURN
      END
```

A2MDIF

```
      SUBROUTINE A2MDIF

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE CALCULATES THE MAXIMUM FIRST DIFFERENCE OF TWO
C     CELL QUANTITIES (DEPENDENT VARIABLES).  THESE VARIABLES ARE
C     POINTED BY K1ADA2 AND K2ADA2.  THE NORMALIZED CELLS VALUES ARE
C     THEN STORED IN CHNGA2 AND WORKA2. SUBSEQUENTLY ONLY NORMALIZED
C     "CHI-SQUARE" VARIABLE IS USED FOR THE DECISION OF ADAPTATION.

C*********************************************************************

C     STEP THROUGH EACH CELL TO ACCUMULATE MAXIMUM AVERAGE DIFFERENCE
C     FOR EACH SPATIAL ADAPTATION CRITERIA VARIABLE

      AVGU1  = 0.
      AVGU2  = 0.
      K3ADA2 = K2ADA2
      IF (K2ADA2 .GT. 100) K3ADA2 = K2ADA2 - 100

      DO 10 JCELL = 1, NPLCA2

C       POINT TO THE ACTUAL CELL

        ICELL = ICELA2(JCELL)

C       SET UP NODE POINTERS FOR THIS CELL

        KSW   = ICELG2(2,ICELL)
        KSE   = ICELG2(4,ICELL)
        KNE   = ICELG2(6,ICELL)
        KNW   = ICELG2(8,ICELL)

C       SAVE DEPENDENT VARIABLES AT ALL CELL CORNERS

        U1SW          = DPENG2(K1ADA2,KSW)
        U1SE          = DPENG2(K1ADA2,KSE)
        U1NE          = DPENG2(K1ADA2,KNE)
        U1NW          = DPENG2(K1ADA2,KNW)
        U1AV          = 0.25*(U1SW + U1SE + U1NE + U1NW)
        U1SW          = U1SW - U1AV
        U1SE          = U1SE - U1AV
        U1NE          = U1NE - U1AV
        U1NW          = U1NW - U1AV
```

```
C         COMPUTE MAXIMUM (OR MINIMUM) FIRST DIFFERENCE FOR THE CELL
          U1MAX          = MAX (U1SW, U1SE, U1NE, U1NW)
          U1MIN          = MIN (U1SW, U1SE, U1NE, U1NW)
          IF (U1MAX .LT. ABS(U1MIN)) U1MAX = U1MIN
          CHNGA2(JCELL) = U1MAX
          AVGU1          = AVGU1 + U1MAX

10      CONTINUE
C
        IF (K3ADA2 .NE. 0) THEN
          DO 20 JCELL = 1, NPLCA2
            ICELL          = ICELA2(JCELL)
            KSW            = ICELG2(2,ICELL)
            KSE            = ICELG2(4,ICELL)
            KNE            = ICELG2(6,ICELL)
            KNW            = ICELG2(8,ICELL)
            U2SW           = DPENG2(K3ADA2,KSW)
            U2SE           = DPENG2(K3ADA2,KSE)
            U2NE           = DPENG2(K3ADA2,KNE)
            U2NW           = DPENG2(K3ADA2,KNW)
            IF (K2ADA2 .GT. 100) THEN
               U2SW = U2SW/DPENG2(1,KSW)
               U2SE = U2SE/DPENG2(1,KSE)
               U2NE = U2NE/DPENG2(1,KNE)
               U2NW = U2NW/DPENG2(1,KNW)
            ENDIF
            U2AV           = 0.25*(U2SW + U2SE + U2NE + U2NW)
            U2SW           = U2SW - U2AV
            U2SE           = U2SE - U2AV
            U2NE           = U2NE - U2AV
            U2NW           = U2NW - U2AV
            U2MAX          = MAX (U2SW, U2SE, U2NE, U2NW)
            U2MIN          = MIN (U2SW, U2SE, U2NE, U2NW)
            IF (U2MAX .LT. ABS(U2MIN)) U2MAX = U2MIN
            WORKA2(JCELL) = U2MAX
            AVGU2          = AVGU2 + U2MAX
20        CONTINUE
        ENDIF

C       COMPUTE THE AVERAGE CHANGE FOR ALL THE CELLS

        AVGU1 = AVGU1/NPLCA2
        AVGU2 = AVGU2/NPLCA2

C       COMPUTE THE VARIANCES OF THESE TWO QUANTITES

        VARU11 = 0.
        VARU12 = 0.
        VARU22 = 0.

        DO 30 JCELL = 1, NPLCA2
          CHNGA2(JCELL) = CHNGA2(JCELL) - AVGU1
          VARU11        = VARU11 + (CHNGA2(JCELL))**2
30      CONTINUE

        IF (K3ADA2 .NE. 0) THEN
          DO 40 JCELL = 1, NPLCA2
```

```
            WORKA2(JCELL) = WORKA2(JCELL) - AVGU2
            VARU12        = VARU12 + WORKA2(JCELL)*CHNGA2(JCELL)
            VARU22        = VARU22 + (WORKA2(JCELL))**2
40        CONTINUE
        ENDIF
C
        IF (NPLCA2 .EQ. 0 .OR. VARU11 .EQ. 0.) THEN
          ZER1 = VARU11
          ZER2 = NPLCA2
           CALL ERRORM (19,'A2MDIF','VARU11',ZER1,'NPLCA2',ZER2,
     1       JPRINT,'STANDARD DEVIATION ERROR')
        ENDIF

        VARU11 = VARU11/NPLCA2
        VARU12 = VARU12/NPLCA2
        VARU22 = VARU22/NPLCA2
        SD1    = SQRT(VARU11)
        SD2    = SQRT(VARU22)

C       COMPUTE THE DETERMINANT OF THE VARIANCE-COVARIANCE MATRIX

        DETERM = VARU11*VARU22 - VARU12*VARU12

C       COMPUTE THE INVERSE OF THE VARIANCE-COVARIANCE MATRIX

        IF (VARU22 .NE. 0.) THEN
           DETINV = 1./DETERM
           DUMMY  = VARU22
           VARU22 = VARU11*DETINV
           VARU11 = DUMMY *DETINV
           VARU12 =-VARU12*DETINV
        ELSE
           VARU11 = 1./VARU11
        ENDIF
C       WRITE THE RESULTS FOR SUBSEQUENT PLOTTING

        IF (KPLTA2 .NE. 0) THEN
           JPLOTA = 61
           WRITE (JPLOTA,1400) NCELA2,VARU11,VARU12,VARU22,AVGU1,AVGU2
           WRITE (JPLOTA,1500) SD1, SD2
           WRITE (JPLOTA,1500)(CHNGA2(NC), WORKA2(NC),NC=1,NPLCA2)
        ENDIF
C                                                                  T  -1
C       REASSIGN THE CHANGE VARIABLES AS AN ONE DIMENSIONAL ARRAY X  S  X

        DO 50 JCELL = 1, NPLCA2
          TERM1 = VARU11*CHNGA2(JCELL)**2
          TERM2 = VARU22*WORKA2(JCELL)**2
          TERM3 = VARU12*WORKA2(JCELL)*CHNGA2(JCELL)
          TERM4 = TERM1 + TERM2 + TERM3
          IF (TERM4 .LT. 0.) THEN
             ZER1 = JCELL
             ZER2 = TERM4
              CALL ERRORM (20,'A2MDIF','JCELL ',ZER1,'TERM4 ',ZER2,
     1         JPRINT,'COVARIANCE MATRIX IS NOT POSITIVE DEFINATE ?')
          ENDIF
          CHNGA2(JCELL) = SQRT(TERM4)
```

```
50      CONTINUE
C          -
C       PRINT OUT PARAMETERS
C
        IF (IDBGA2 .NE. 8 .AND. IDBGA2 .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) AVGU1, AVGU2, VARU11, VARU12, VARU22
        WRITE(JDEBUG,1500) (CHNGA2(I), I = 1, NPLCA2)

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM A2MDIF' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'AVGU1  =',G14.5,5X,'AVGU2  =',G14.5/5X,
     1     'VARU11 =',G14.5,5X,'VARU12 =',G14.5,5X,'VARU22 =',G14.5,
     2     /10X,'CHANGES AFTER NORMALIZATION')
1400    FORMAT(I7,7G14.5)
1500    FORMAT(8G14.5)

        RETURN
        END
```

A2MTHU

```
        SUBROUTINE A2MTHO
C                  A2MTHU

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'

C*********************************************************************

C       THIS SUBROUTINE CALLS ALL THE OTHER ROUTINES USED FOR SPATIAL
C       ADAPTATION,  THUS WHENEVER SPATIAL ADAPTATION IS NEEDED ONLY
C       THIS ROUTINE MUST BE CALLED.

C*********************************************************************

C       INITIALIZE THE NUMBER OF PLACES WHERE CALCULATIONS WILL BE DONE

        NPLCA2 = NCELA2

C       CALL THE PARTICULAR KIND OF METHOD

        CALL A2MDIF
```

```
C       FIND THE THRESHOLD VALUES
        CALL A2THRS

C       FINALLY CALL THE ADAPTIVE GRID ROUTINE
        CALL A2ADPO

C       IF THERE ARE VOIDS IN THE SPATIAL GRIDS, REMOVE THEM
        CALL A2VOID

C       RESET THE CEWIC CELL ARRAY POINTER

        CALL A2CEWC

        RETURN
        END
```

A2MTH0

```
        SUBROUTINE A2MTH0

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] A2COMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'

C***********************************************************************

C       THIS SUBROUTINE DECIDES UPON THE METHOD OF COMPUTING CHANGES
C       USED IN THE DECISION PROCESS OF SPATIAL ADAPTATION.  IF THE
C       PARAMETER MTYPA2 IS EVEN THEN THE CALCULATION IS CELL BASED
C       AND WHEN IT IS ODD THE CALCULATION IS NODE BASED.  IT ALSO
C       CALLS ALL THE OTHER ROUTINES USED FOR SPATIAL ADAPTATION,
C       THUS WHENEVER SPATIAL ADAPTATION IS NEEDED ONLY THIS ROUTINE
C       MUST BE CALLED.

C***********************************************************************

C       CHECK IF YOU WANT TO SKIP THE ADAPTATION ALTOGETHER
        IF (K1ADA2 .EQ. 0) RETURN

C       INITIALIZE THE NUMBER OF PLACES WHERE CALCULATIONS WILL BE DONE

        IF (MTYPA2 .EQ. 0) THEN
          NPLCA2 = NCELA2
        ELSE
          NPLCA2 = NNODG2
        ENDIF

C       CALL THE PARTICULAR KIND OF METHOD

        IF (METHA2 .EQ. 1) CALL A2VALN
        IF (METHA2 .EQ. 2) CALL A2VALC
        IF (METHA2 .EQ. 3) CALL A2GRDN
```

```
      IF (METHA2 .EQ. 4) CALL A2GRDC
C     IF (METHA2 .EQ. 5) CALL A2LAPL
      IF (METHA2 .EQ. 6) CALL A2MDIF

C     FIND THE THRESHOLD VALUES
      CALL A2THRS

C     FINALLY CALL THE ADAPTIVE GRID ROUTINE
      CALL A2ADPO

C     IF THERE ARE VOIDS IN THE SPATIAL GRIDS, REMOVE THEM
      CALL A2VOID

C     RESET THE CEWIC CELL ARRAY POINTER

      CALL A2CEWC
C
C     PRINT OUT PARAMETERS
C
      IF (IDBGA2 .NE. 2 .AND. IDBGA2 .LT. 1000)  RETURN

      WRITE(JDEBUG,1000)
      WRITE(JDEBUG,1100)
      WRITE(JDEBUG,1200)
      WRITE(JDEBUG,1300) K1ADA2, K2ADA2, NPLCA2, NCELA2, MTYPA2, METHA2

C     -----------------
C     FORMAT STATEMENTS
C     -----------------

1000  FORMAT(//10X,'----------------------' )
1100  FORMAT(  10X,'DEBUG PRINT FROM A2MTHO' )
1200  FORMAT(  10X,'----------------------'/)
1300  FORMAT( 5X,'SPATIAL ADAPTATION  CRITERIA 1 =',I5,
     1        10X,'SPATIAL ADAPTATION  CRITERIA 2 =',I5/
     2         5X,'NO. PLACES FOR DATA COLLECTION =',I5,
     4        10X,'NUMBER OF CEWIC CELLS          = ',I5/
     1         5X,'CALCULATION BASIS (CELL: EVEN) =',I5,
     6        10X,'METHOD OF VARIATION CALCULATION=',I5)

      RETURN
      END
```

A2PLOT

```
      PROGRAM A2PLOT

      INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] A2COMN.INC/LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] IOCOMN.INC/LIST'

      PARAMETER (NBIN=201)
```

```
      DIMENSION THRESH(NBIN)  , FRACTN(NBIN)
      DIMENSION IN$(3)        , IOPT$(3)
      CHARACTER PLTITL*80     , YESNO*1
      REAL*4    E1XAX$(MCELG2), E1YAX$(MCELG2)

C*********************************************************************

C       THIS SUBROUTINE PLOTS THE FOLLOWING CURVES :-
C           1. VARIATION 1 VS VARIATION 2
C           2. REFINEMENT PARAMETER VS FREQUENCY
C       THE DATA WAS WRITTEN BY SUBROUTINES A2MDIF AND A2THRS ON
C       A SEQUENTIAL FILE.

C*********************************************************************

      JPLOTA   = 61
      JTERMI   = 5
      JTERMO   = 6
      IOPT$(1) = 12

C       READ THE PREVIOUSLY WRITTEN DATA

      OPEN (UNIT=JPLOTA, FILE='APLOTS.DAT', STATUS='OLD' )

800   FORMAT(I7,7G14.5)
900   FORMAT(8G14.5)

      READ (JPLOTA,800) NCELA2,VARU11,VARU12,VARU22,AVGU1,AVGU2
      READ (JPLOTA,900) SD1, SD2
      READ (JPLOTA,900)(CHNGA2(NC), WORKA2(NC),NC=1,NCELA2)
      READ (JPLOTA,800) MBIN, THRDA2, THRCA2
      READ (JPLOTA,900) (THRESH(IBIN),FRACTN(IBIN),IBIN=1,MBIN)
      CLOSE(JPLOTA)

      DENOX = 1.
      DENOY = 1.
      IF (SD1 .NE. 0.) DENOX=1./SD1
      IF (SD2 .NE. 0.) DENOY=1./SD2

      WRITE (JTERMO,1000)
1000  FORMAT(/1X,'INPUT THE MAIN TITLE ')
1100  FORMAT(A1)
      READ (JTERMI,1200) MTITLE
1200  FORMAT(A80)
      CALL GR_INIT (JTERMI, JTERMO, MTITLE)

5     WRITE (JTERMO,*) ' 1. NCELA2 =',NCELA2
      WRITE (JTERMO,*) ' 2. VARU11 =',VARU11
      WRITE (JTERMO,*) ' 3. VARU12 =',VARU12
      WRITE (JTERMO,*) ' 4. VARU22 =',VARU22
      WRITE (JTERMO,*) ' 5. THRDA2 =',THRDA2
      WRITE (JTERMO,*) ' 6. THRCA2 =',THRCA2
      WRITE (JTERMO,*) ' 7. AVGU1  =',AVGU1
      WRITE (JTERMO,*) ' 8. AVGU2  =',AVGU2
      WRITE (JTERMO,*) ' 9. DENOX  =',DENOX
      WRITE (JTERMO,*) '10. DENOY  =',DENOY
      WRITE (JTERMO,*) ' Give the number of value to be changed'
```

```
      READ (JTERMI,*) IVALUE
      IF (IVALUE .EQ. 3) READ (JTERMI,*) VARU12
      IF (IVALUE .EQ. 5) READ (JTERMI,*) THRDA2
      IF (IVALUE .EQ. 9) READ (JTERMI,*) DENOX
      IF (IVALUE .EQ.10) READ (JTERMI,*) DENOY
      IF (IVALUE .NE. 0) GOTO 5

      WRITE (JTERMO,1300)
1300  FORMAT(1X,'DO YOU WANT TO PLOT THE VARIATION1/VARIATION2 PLOT')

      READ(JTERMI,1100) YESNO
      IF (YESNO.EQ.'n' .OR. YESNO.EQ.'N') GOTO 30

      INLIN    = 3
      IN$(1)    = NCELA2
      IN$(2)    = 2*MBIN
      IN$(3)    = 2*MBIN
c     PLTITL = 'VARIABLE1~VARIABLE2~ DISTRIBUTION OF VARIATIONS'
      PLTITL(1:42)  = 'DENSITY VARIATION~MASS FRACTION VARIATION~'
      PLTITL(43:80) = 'DISTRIBUTION OF VARIATIONS'

      DO 10 I = 1, NCELA2
        E1XAX$(I) = CHNGA2(I)*denox
        E1YAX$(I) = WORKA2(I)*denoy
10    CONTINUE

      DISCRI = VARU11*VARU22 - VARU12**2

      IF (DISCRI .LE. 0.) THEN
         INLIN = 1
      ELSE
         X2MAX  = THRDA2*SQRT(VARU11/DISCRI)
         X2MIN  =-X2MAX
         DX2    = (X2MAX-X2MIN)/MBIN
         C2S11  = THRDA2**2*VARU11
         RVAR11 = 1./VARU11
         IBEG   = NCELA2
         IEND   = NCELA2 + 2*MBIN + 1
         IBG2   = NCELA2 + 2*MBIN
         IED2   = NCELA2 + 4*MBIN + 1
         DO 20 I = 1, MBIN
            X2             = X2MIN + (I-1)*DX2
            DD             = C2S11/(X2*X2) - DISCRI
c           DD             = SQRT(DD)
            DD             = SQRT(abs(DD))
            DD1            = -VARU12 + DD
            DD2            = -VARU12 - DD
            X2RVAR         = X2*RVAR11
            E1XAX$(IBEG+I) = DD1*X2RVAR*denox
            E1XAX$(IEND-I) = DD2*X2RVAR*denox
            E1YAX$(IBEG+I) = X2*denoy
            E1YAX$(IEND-I) = X2*denoy

            E1XAX$(IBG2+I) = E1XAX$(IBEG+I)*THRCA2
            E1XAX$(IED2-I) = E1XAX$(IEND-I)*THRCA2
            E1YAX$(IBG2+I) = E1YAX$(IBEG+I)*THRCA2
            E1YAX$(IED2-I) = E1YAX$(IEND-I)*THRCA2
```

```
20          CONTINUE
        ENDIF

        INDGR    = 21
        IOPT$(2) = 10
        IOPT$(3) = 10

        CALL GR_LINE(IOPT$,INLIN,PLTITL,INDGR,E1XAX$,E1YAX$,IN$)

30      WRITE (JTERMO,1400)
1400    FORMAT(1X,'DO YOU WANT TO PLOT THRESHOLD/FREQUENCY [Y,N,Z]')

        READ(JTERMI,1100) YESNO
        IF (YESNO.EQ.'z' .OR. YESNO.EQ.'Z') GOTO 5
        IF (YESNO.EQ.'n' .OR. YESNO.EQ.'N') STOP

        DO 40 I = 1, MBIN
          E1XAX$(I) = THRESH(I)
          E1YAX$(I) = FRACTN(I)
40      CONTINUE

        PLTITL = 'REFINEMENT PARAMETER~FREQUENCY~ CUMULATIVE FREQUENCY'
        INLIN    = 1
        INDGR    = 21
        IOPT$(1) = 14
        IN$(1)   = MBIN
        WRITE (JTERMO,1500) THRDA2, THRCA2
1500    FORMAT(5X,'THRDA2 =',G15.5,5X,'THRCA2 =',G15.5)

        CALL PLXSET(IOPT$,INDGR)

        CALL GR_LINE(IOPT$,INLIN,PLTITL,INDGR,E1XAX$,E1YAX$,IN$)

        END
```

A2THRU

```
        SUBROUTINE A2THRS
C                  A2THRU

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        PARAMETER (NBIN=201)
        DIMENSION THRESH(NBIN), FRACTN(NBIN)
        DATA KOUNT /0/
        SAVE KOUNT

C*****************************************************************

C       THIS SUBROUTINE PICKS THE DIVIDE THRESHOLD (THRDA2) AND
C          COLLAPSE THRESHOLD (THRCA2) SUCH THAT:
```

```
C          -   THRDA2 = MAX (THRSH1, ALPHA2)
C          -_  THRCA2 = MIN (THRSH2, GAMMA2)

C*********************************************************************

        KOUNT  = KOUNT + 1
        IF (KOUNT .GT. MTHRA2) KOUNT = 1
        IF (KOUNT .NE. 1     ) RETURN
C
C       COMPUTE THE MAXIMUM VARIATION, ASSUUMING THAT THE MINIMUM
C       ONE IS ZERO. (AVERAGE VARIATION IS ABOUT 1.2 )

        THRMIN =  0.
        THRMAX =  0.

        DO 10 IPLAC = 1, NPLCA2
          THRMAX = MAX(THRMAX,CHNGA2(IPLAC))
10      CONTINUE

        DTHRSH =  (THRMAX-THRMIN)/(NBIN-1)

C       INITIALIZE THRESH AND FRACTN BIN VALUES

        DO 20 IBIN = 1, NBIN
          THRESH(IBIN) = THRMIN + (IBIN-1)*DTHRSH
          FRACTN(IBIN) = 0.
20      CONTINUE

C       ACCUMULATE NUMBER OF POINTS IN EACH THRESHOLD BIN

        DO 30 IPLAC = 1, NPLCA2
          IBIN         = 1 + INT(CHNGA2(IPLAC)/DTHRSH)
          FRACTN(IBIN) = FRACTN(IBIN) + 1.0
30      CONTINUE
C
C       CALCULATE (CUMMULATIVE) ACTUAL FRACTION OF POINTS WITH
C       VARIATION BELOW THRESH

        SUM = 0.
        DO 40 IBIN = NBIN, 1, -1
          SUM          = SUM + FRACTN(IBIN)
          FRACTN(IBIN) = SUM/NPLCA2
40      CONTINUE
C
C       FIND THRESHOLD POINT WHERE CUMMULATIVE FRACTION EQUALS BETAA2
C       IF, E.G., BETAA2=0.2, THEN ATMOST 20% CELLS CAN BE ADAPTED

        DO 50 IBIN = NBIN, 1, -1
          IF(FRACTN(IBIN) .GT. BETAA2) THEN
             THRSH1 = THRESH(IBIN+1)
             GOTO 60
          ENDIF
50      CONTINUE

C       DETERMINE DELETE THRESHOLD CRITERIA

60      THRDA2 = MAX (THRSH1, ALPHA2)
```

```
C       FIND COLLAPSE THRESHOLD CRITERIA AS A SPECIFIED PERCENTAGE OF
C       THE MAXIMUM THRESHOLD CRITERIA, A TYPICAL VALUE IS 20%

        THRSH2 = GAMMA2*THRDA2

C       DETERMINE COLLAPSE THRESHOLD CRITERIA

        THRCA2 = MIN(THRSH2,DELTA2)

        RETURN
        END
```

A2THRS

```
        SUBROUTINE A2THRS

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] A2COMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        PARAMETER (NBIN=201)
        DIMENSION THRESH(NBIN), FRACTN(NBIN)
        DATA KOUNT /0/
        SAVE KOUNT

C***********************************************************************

C       THIS SUBROUTINE PICKS THE DIVIDE THRESHOLD (THRDA2) AND
C          COLLAPSE THRESHOLD (THRCA2) SUCH THAT:
C          -   THRDA2 = MAX (THRSH1, ALPHA2)
C          -   THRCA2 = MIN (THRSH2, GAMMA2)

C***********************************************************************

        KOUNT  = KOUNT + 1
        IF (KOUNT .GT. MTHRA2) KOUNT = 1
        IF (KOUNT .NE. 1     ) RETURN
C
C       COMPUTE THE MAXIMUM VARIATION, ASSUUMING THAT THE MINIMUM
C       ONE IS ZERO. (AVERAGE VARIATION IS ABOUT 1.2 )

        THRMIN =  0.
        THRMAX =  0.

        DO 10 IPLAC = 1, NPLCA2
          THRMAX = MAX(THRMAX,CHNGA2(IPLAC))
10      CONTINUE

        DTHRSH =  (THRMAX-THRMIN)/(NBIN-1)

C       INITIALIZE THRESH AND FRACTN BIN VALUES
```

```
      DO 20 IBIN = 1, NBIN
        THRESH(IBIN) = THRMIN + (IBIN-1)*DTHRSH
        FRACTN(IBIN) = 0.
20    CONTINUE

C     ACCUMULATE NUMBER OF POINTS IN EACH THRESHOLD BIN

      DO 30 IPLAC = 1, NPLCA2
        IBIN         = 1 + INT(CHNGA2(IPLAC)/DTHRSH)
        FRACTN(IBIN) = FRACTN(IBIN) + 1.0
30    CONTINUE
C
C     CALCULATE (CUMMULATIVE) ACTUAL FRACTION OF POINTS WITH
C     VARIATION BELOW THRESH

      SUM = 0.
      DO 40 IBIN = NBIN, 1, -1
        SUM          = SUM + FRACTN(IBIN)
        FRACTN(IBIN) = SUM/NPLCA2
40    CONTINUE
C
C     FIND THRESHOLD POINT WHERE CUMMULATIVE FRACTION EQUALS BETAA2
C     IF, E.G., BETAA2=0.2, THEN ATMOST 20% CELLS CAN BE ADAPTED

      DO 50 IBIN = NBIN, 1, -1
        IF(FRACTN(IBIN) .GT. BETAA2) THEN
           THRSH1 = THRESH(IBIN+1)
           GOTO 60
        ENDIF
50    CONTINUE

C     DETERMINE DELETE THRESHOLD CRITERIA

60    THRDA2 = MAX (THRSH1, ALPHA2)

C     FIND COLLAPSE THRESHOLD CRITERIA AS A SPECIFIED PERCENTAGE OF
C     THE MAXIMUM THRESHOLD CRITERIA, A TYPICAL VALUE IS 20%

      THRSH2 = GAMMA2*THRDA2

C     DETERMINE COLLAPSE THRESHOLD CRITERIA

      THRCA2 = MIN(THRSH2,DELTA2)

C     SEE IF THE DATA FOR PLOTTING THRESHOLD GRAPHS IS NEEDED

      IF (KPLTA2 .NE. 0) THEN
         JPLOTA = 61
         WRITE(JPLOTA,800) NBIN, THRDA2, THRCA2
         WRITE(JPLOTA,900) (THRESH(IBIN),FRACTN(IBIN),IBIN=1,NBIN)
         CLOSE(JPLOTA)
      ENDIF
C
C     PRINT OUT PARAMETERS
C
      IF (IDBGA2 .NE. 9 .AND. IDBGA2 .LT. 1000)  RETURN
```

```
      WRITE(JDEBUG,1000)
      WRITE(JDEBUG,1100)
      WRITE(JDEBUG,1200)
      WRITE(JDEBUG,1300) ALPHA2, BETAA2, GAMMA2, DELTA2,
     1                   THRSH1, THRSH2, THRDA2, THRCA2
      WRITE(JDEBUG,1400) THRMAX, DTHRSH, MTHRA2, KPLTA2
      WRITE(JDEBUG,1500)

      DO 70 IBIN = 1, NBIN
         WRITE(JDEBUG,1600) IBIN, THRESH(IBIN), FRACTN(IBIN)
70    CONTINUE

C     -----------------
C     FORMAT STATEMENTS
C     -----------------
C
800   FORMAT(I7,7G14.5)
900   FORMAT(8G14.5)
1000  FORMAT(//10X,'-----------------------' )
1100  FORMAT(  10X,'DEBUG PRINT FROM A2THRS' )
1200  FORMAT(  10X,'-----------------------'/)
1300  FORMAT(9X,'ALPHA2',8X,'BETAA2',8X,'GAMMA2',8X,'DELTA2',8X,
     1          'THRSH1',8X,'THRSH2',8X,'THRDA2',8X,'THRCA2'/5X,
     2          8G14.5)
1400  FORMAT(9X,'THRMAX',8X,'DTHRSH',8X,'MTHRA2',8X,'KPLTA2'/5X,
     1          2G14.5,3X,I5,8X,I5)
1500  FORMAT(/12X,'IBIN',4X,'THRESH',7X,'FRACTN')
1600  FORMAT(10X,I5,2G14.5)

      RETURN
      END
```

A2VALC

```
      SUBROUTINE A2VALC

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE ASSIGNS THE CELL VALUES OF TWO QUANTITIES
C     (DEPENDENT VARIABLES) TO THE SPATIAL ADAPTATION CRITERIA
C     VARIABLES WHICH ARE POINTED BY K1ADA2 AND K2ADA2.  THE NORMALIZED
C     CELL VALUES ARE THEN STORED IN CHNGA2 AND WORKA2. SUBSEQUENTLY
C     ONLY NORMALIZED "CHI-SQUARE" VARIABLE IS USED FOR THE DECISION
C     OF SPATIAL ADAPTATION.

C*********************************************************************
C
```

```
C       STEP THROUGH EACH CEWIC CELL TO COLLECT VALUES FOR FIRST
C       SPATIAL ADAPTATION CRITERIA VARIABLE

        DO 10 IPLAC = 1, NPLCA2

C         POINT TO THE ACTUAL CELL

          ICELL = ICELA2(IPLAC)

C         SET UP NODE POINTERS FOR THIS CELL

          KSW  = ICELG2(2,ICELL)
          KSE  = ICELG2(4,ICELL)
          KNE  = ICELG2(6,ICELL)
          KNW  = ICELG2(8,ICELL)

C         SAVE DEPENDENT VARIABLES AT ALL CELL CORNERS

          U1SW = DPENG2(K1ADA2,KSW)
          U1SE = DPENG2(K1ADA2,KSE)
          U1NE = DPENG2(K1ADA2,KNE)
          U1NW = DPENG2(K1ADA2,KNW)

C         COMPUTE THE AVERAGE VALUE AT EACH CELL

          CHNGA2(IPLAC) = 0.25*(U1SW + U1SE + U1NE + U1NW)

10      CONTINUE
C
C       NOW CHECK IF THE SECOND CRITERIA VARIABLE EXISTS
C
        IF (K2ADA2 .NE. 0) THEN
          DO 20 IPLAC = 1, NPLCA2
            ICELL = ICELA2(IPLAC)
            KSW   = ICELG2(2,ICELL)
            KSE   = ICELG2(4,ICELL)
            KNE   = ICELG2(6,ICELL)
            KNW   = ICELG2(8,ICELL)
            U2SW  = DPENG2(K2ADA2,KSW)
            U2SE  = DPENG2(K2ADA2,KSE)
            U2NE  = DPENG2(K2ADA2,KNE)
            U2NW  = DPENG2(K2ADA2,KNW)
            WORKA2(IPLAC) = 0.25*(U2SW + U2SE + U2NE + U2NW)
20        CONTINUE
        ENDIF

C       COMPUTE THE AVERAGE CHANGE FOR ALL THE CELLS

        AVGU1  = 0.
        AVGU2  = 0.

        DO 30 IPLAC = 1, NPLCA2
          AVGU1 = AVGU1 + CHNGA2(IPLAC)
          AVGU2 = AVGU2 + WORKA2(IPLAC)
30      CONTINUE
C
        AVGU1 = AVGU1/NPLCA2
```

```
      AVGU2 = AVGU2/NPLCA2
C
C     COMPUTE THE VARIANCES OF THESE TWO QUANTITES
C
      VARU11 = 0.
      VARU12 = 0.
      VARU22 = 0.
C
      DO 40 IPLAC = 1, NPLCA2
        CHNGA2(IPLAC) = CHNGA2(IPLAC) - AVGU1
        VARU11        = VARU11 + (CHNGA2(IPLAC))**2
40    CONTINUE
C
      IF (K2ADA2 .NE. 0) THEN
        DO 50 IPLAC = 1, NPLCA2
          WORKA2(IPLAC) = WORKA2(IPLAC) - AVGU2
          VARU12        = VARU12 + WORKA2(IPLAC)*CHNGA2(IPLAC)
          VARU22        = VARU22 + (WORKA2(IPLAC))**2
50      CONTINUE
      ENDIF
C
      IF (NPLCA2 .EQ. 0 .OR. VARU11 .EQ. 0.) THEN
        ZER1 = VARU11
        ZER2 = NPLCA2
         CALL ERRORM (19,'A2VALC','VARU11',ZER1,'NPLCA2',ZER2,
     1             JPRINT,'STANDARD DEVIATION ERROR')
      ENDIF
C
      VARU11 = VARU11/NPLCA2
      VARU12 = VARU12/NPLCA2
      VARU22 = VARU22/NPLCA2

C     COMPUTE THE DETERMINANT OF THE VARIANCE-COVARIANCE MATRIX

      DETERM = VARU11*VARU22 - VARU12*VARU12

C     COMPUTE THE INVERSE OF THE VARIANCE-COVARIANCE MATRIX

      IF (VARU22 .NE. 0.) THEN
         DETINV = 1./DETERM
         DUMMY  = VARU22
         VARU22 = VARU11*DETINV
         VARU11 = DUMMY *DETINV
         VARU12 =-VARU12*DETINV
      ELSE
         VARU11 = 1./VARU11
      ENDIF

C     WRITE THE RESULTS FOR SUBSEQUENT PLOTTING

      IF (KPLTA2 .NE. 0) THEN
         JPLOTA = 61
         WRITE (JPLOTA,1400) NCELA2,VARU11,VARU12,VARU22,AVGU1,AVGU2
         WRITE (JPLOTA,1500)(CHNGA2(NC), WORKA2(NC),NC=1,NPLCA2)
      ENDIF
C                                                                  T  -1
C     REASSIGN THE CHANGE VARIABLES AS AN ONE DIMENSIONAL ARRAY X  S  X
```

```
      DO 60 IPLAC = 1, NPLCA2
        TERM1 = VARU11*CHNGA2(IPLAC)**2
        TERM2 = VARU22*WORKA2(IPLAC)**2
        TERM3 = VARU12*WORKA2(IPLAC)*CHNGA2(IPLAC)
        TERM4 = TERM1 + TERM2 + TERM3
        IF (TERM4 .LT. 0.) THEN
           ZER1 = IPLAC
           ZER2 = TERM4
            CALL ERRORM (20,'A2VALC','IPLAC ',ZER1,'TERM4 ',ZER2,
     1        JPRINT,'COVARIANCE MATRIX IS NOT POSITIVE DEFINATE ?')
        ENDIF
        CHNGA2(IPLAC) = SQRT(TERM4)
60    CONTINUE
C
C     PRINT OUT PARAMETERS
C
      IF (IDBGA2 .NE. 4 .AND. IDBGA2 .LT. 1000)  RETURN

      WRITE(JDEBUG,1000)
      WRITE(JDEBUG,1100)
      WRITE(JDEBUG,1200)
      WRITE(JDEBUG,1300) AVGU1, AVGU2, VARU11, VARU12, VARU22
      WRITE(JDEBUG,1500) (CHNGA2(I), I = 1, NPLCA2)

C     -----------------
C     FORMAT STATEMENTS
C     -----------------

1000  FORMAT(//10X,'----------------------' )
1100  FORMAT(  10X,'DEBUG PRINT FROM A2VALC' )
1200  FORMAT(  10X,'----------------------'/)
1300  FORMAT(5X,'AVGU1  =',G14.5,5X,'AVGU2  =',G14.5/5X,
     1     'VARU11 =',G14.5,5X,'VARU12 =',G14.5,5X,'VARU22 =',G14.5,
     2     /10X,'CHANGES AFTER NORMALIZATION')
1400  FORMAT(I7,7G14.5)
1500  FORMAT(8G14.5)

      RETURN
      END
```

A2VALN

```
      SUBROUTINE A2VALN

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'

C*********************************************************************
```

```
C       THIS SUBROUTINE ASSIGNS THE NODAL VALUES OF TWO QUANTITIES
C       (DEPENDENT VARIABLES) TO THE SPATIAL ADAPTATION CRITERIA
C       VARIABLES WHICH ARE POINTED BY K1ADA2 AND K2ADA2.  THE NORMALIZED
C       NODE VALUES ARE THEN STORED IN CHNGA2 AND WORKA2. SUBSEQUENTLY
C       ONLY NORMALIZED "CHI-SQUARE" VARIABLE IS USED FOR THE DECISION
C       OF SPATIAL ADAPTATION.

C***********************************************************************
C
C       STEP THROUGH EACH NODE TO COLLECT VALUES FOR FIRST SPATIAL
C       ADAPTATION CRITERIA VARIABLE

        DO 10 IPLAC = 1, NPLCA2
          CHNGA2(IPLAC) = DPENG2(K1ADA2,IPLAC)
10      CONTINUE
C
C       NOW CHECK IF THE SECOND CRITERIA VARIABLE EXISTS
C
        IF (K2ADA2 .NE. 0) THEN
          DO 20 IPLAC = 1, NPLCA2
            WORKA2(IPLAC) = DPENG2(K2ADA2,IPLAC)
20        CONTINUE
        ENDIF

C       COMPUTE THE AVERAGE CHANGE FOR ALL THE NODES

        AVGU1  = 0.
        AVGU2  = 0.

        DO 30 IPLAC = 1, NPLCA2
          AVGU1 = AVGU1 + CHNGA2(IPLAC)
          AVGU2 = AVGU2 + WORKA2(IPLAC)
30      CONTINUE
C
        AVGU1 = AVGU1/NPLCA2
        AVGU2 = AVGU2/NPLCA2
C
C       COMPUTE THE VARIANCES OF THESE TWO QUANTITES
C
        VARU11 = 0.
        VARU12 = 0.
        VARU22 = 0.
C
        DO 40 IPLAC = 1, NPLCA2
          CHNGA2(IPLAC) = CHNGA2(IPLAC) - AVGU1
          VARU11        = VARU11 + (CHNGA2(IPLAC))**2
40      CONTINUE
C
        IF (K2ADA2 .NE. 0) THEN
          DO 50 IPLAC = 1, NPLCA2
            WORKA2(IPLAC) = WORKA2(IPLAC) - AVGU2
            VARU12        = VARU12 + WORKA2(IPLAC)*CHNGA2(IPLAC)
            VARU22        = VARU22 + (WORKA2(IPLAC))**2
50        CONTINUE
        ENDIF

        IF (NPLCA2 .EQ. 0 .OR. VARU11 .EQ. 0.) THEN
```

```
          ZER1 = VARU11
          ZER2 = NPLCA2
           CALL ERRORM (19,'A2VALN','VARU11',ZER1,'NPLCA2',ZER2,
     1             JPRINT,'STANDARD DEVIATION ERROR')
          ENDIF
C
          VARU11 = VARU11/NPLCA2
          VARU12 = VARU12/NPLCA2
          VARU22 = VARU22/NPLCA2

C         COMPUTE THE DETERMINANT OF THE VARIANCE-COVARIANCE MATRIX

          DETERM = VARU11*VARU22 - VARU12*VARU12

C         COMPUTE THE INVERSE OF THE VARIANCE-COVARIANCE MATRIX

          IF (VARU22 .NE. 0.) THEN
             DETINV = 1./DETERM
             DUMMY  = VARU22
             VARU22 = VARU11*DETINV
             VARU11 = DUMMY *DETINV
             VARU12 =-VARU12*DETINV
          ELSE
             VARU11 = 1./VARU11
          ENDIF

C         WRITE THE RESULTS FOR SUBSEQUENT PLOTTING

          IF (KPLTA2 .NE. 0) THEN
             JPLOTA = 61
             WRITE (JPLOTA,1400) NCELA2,VARU11,VARU12,VARU22,AVGU1,AVGU2
             WRITE (JPLOTA,1500)(CHNGA2(NC), WORKA2(NC),NC=1,NPLCA2)
          ENDIF
C                                                                    T  -1
C         REASSIGN THE CHANGE VARIABLES AS AN ONE DIMENSIONAL ARRAY X  S  X

          DO 60 IPLAC = 1, NPLCA2
            TERM1 = VARU11*CHNGA2(IPLAC)**2
            TERM2 = VARU22*WORKA2(IPLAC)**2
            TERM3 = VARU12*WORKA2(IPLAC)*CHNGA2(IPLAC)
            TERM4 = TERM1 + TERM2 + TERM3
            IF (TERM4 .LT. 0.) THEN
               ZER1 = IPLAC
               ZER2 = TERM4
                CALL ERRORM (20,'A2VALN','IPLAC ',ZER1,'TERM4 ',ZER2,
     1              JPRINT,'COVARIANCE MATRIX IS NOT POSITIVE DEFINATE ?')
            ENDIF
            CHNGA2(IPLAC) = SQRT(TERM4)
60        CONTINUE
C
C         PRINT OUT PARAMETERS
C
          IF (IDBGA2 .NE. 3 .AND. IDBGA2 .LT. 1000)  RETURN

          WRITE(JDEBUG,1000)
          WRITE(JDEBUG,1100)
          WRITE(JDEBUG,1200)
```

```
      WRITE(JDEBUG,1300) AVGU1, AVGU2, VARU11, VARU12, VARU22
      WRITE(JDEBUG,1500) (CHNGA2(I), I = 1, NPLCA2)

C     -----------------
C     FORMAT STATEMENTS
C     -----------------

1000  FORMAT(//10X,'----------------------' )
1100  FORMAT(  10X,'DEBUG PRINT FROM A2VALN' )
1200  FORMAT(  10X,'----------------------'/)
1300  FORMAT(5X,'AVGU1  =',G14.5,5X,'AVGU2  =',G14.5/5X,
     1     'VARU11 =',G14.5,5X,'VARU12 =',G14.5,5X,'VARU22 =',G14.5,
     2     /10X,'CHANGES AFTER NORMALIZATION')
1400  FORMAT(I7,7G14.5)
1500  FORMAT(8G14.5)

      RETURN
      END
```

A2VOUU

```
      SUBROUTINE A2VOID
C                 A2VOUU
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'HEXCOD.INC'

      DIMENSION INB(12)

C**********************************************************************

C     THIS SUBROUTINE DETECTS THE VOID CELLS AFTER THE PREVIOUS
C     ADAPTATION CYCLE OF GRID DIVISION, EXTENSION AND COLLAPSE.
C     A VOIDS CELL IS THE ONE WITH ONE OF THE FOLLOWING PROPERTIES:
C       0. HAS FOUR DIVIDED EDGES
C       1. HAS THREE DIVIDED EDGES
C       2. HAS TWO DIVIDED EDGES ON A BOUNDARY
C     THE CELLS WITH FOUR DIVIDED EDGES NEED NOT BE STORED FOR
C     SUBSEQUENT CHECKING OF NEIGHBOURS FOR VOID CELLS.
C     ONLY THE CEWIC CELLS NEED BE CHECKED FOR VOIDS AND ISLANDS.
C     NVOID : THE NUMBER OF CELLS WHICH ARE DETECTED TO BE VOID
C             CELLS (STORED IN MRKDA2 IN PASS 1 AND IN MRKCA2 IN
C             PASS 2)
C
C     THE ISLAND CELLS ARE DEFINED TO ONE OF THE FOLLOWING
C       0. HAVE FOUR DIVIDED EDGES
C       1. HAVE THREE DIVIDED EDGES
C     NVOID : THE NUMBER OF CELLS WHICH ARE DETECTED TO BE ISLAND
C             CELLS (STORED IN MRKDA2 IN PASS 1 AND IN MRKCA2 IN
C             PASS 2)
```

```
C*********************************************************************
C
C
C       + + + + + + + + + + + + + + + + +
C       + NBNWC | NBNWH + NBNEH | NBNEC +
C       +       |       +       |       +
C       +-------+-------+-------+-------+
C       +       |KNW   KN    KNE|       +
C       + NBNWV |               | NBNEV +
C       + + + + +KW   ICELL   KE+ + + + +
C       + NBSWV |               | NBSEV +
C       +       |KSW   KS    KSE|       +
C       +-------+-------+-------+-------+
C       +       |       +       |       +
C       + NBSWC | NBSWH + NBSEH | NBSEC +
C       + + + + + + + + + + + + + + + + +
C
C
C       INITIALIZE THE NUMBER OF VOID CELLS, NDEDGE INDICATES THE
C       NUMBER OF DIVIDED EDGES

        NVOID  = 0
        NTIME  = 1
        IWARN  = 0
C
C       CHECK FOR ISLANDS
C
        DO 190 ICELL = 1, NCELG2

C          ****   CHECK_CELL_CENTER
           IF (ICELG2(1,ICELL) .NE. 0) THEN
              NDEDGE = 0
              KC     = ICELG2(1,ICELL)
              KS     = ICELG2(3,ICELL)
              KE     = ICELG2(5,ICELL)
              KN     = ICELG2(7,ICELL)
              KW     = ICELG2(9,ICELL)
              NBSWH  = NEIBG2(1,KS )
              NBSEH  = NEIBG2(2,KS )
              NBSEV  = NEIBG2(2,KE )
              NBNEV  = NEIBG2(3,KE )
              NBNEH  = NEIBG2(3,KN )
              NBNWH  = NEIBG2(4,KN )
              NBNWV  = NEIBG2(4,KW )
              NBSWV  = NEIBG2(1,KW )
C
              IF (NBSWH .EQ. NBSEH) NDEDGE = NDEDGE + 1
              IF (NBSEV .EQ. NBNEV) NDEDGE = NDEDGE + 1
              IF (NBNEH .EQ. NBNWH) NDEDGE = NDEDGE + 1
              IF (NBSWV .EQ. NBNWV) NDEDGE = NDEDGE + 1
C
C             ####   CHECK_FOR_COLLAPSE
              IF (NDEDGE .GE. 3) THEN
                NCSW = NEIBG2(1,KC)
                NCSE = NEIBG2(2,KC)
                NCNE = NEIBG2(3,KC)
                NCNW = NEIBG2(4,KC)
```

```
                ISUP = ICELG2(10,NCSW)
C
C           -   %%%%   VERIFY_SUPERCELL
                IF (ISUP .EQ. ICELL) THEN
                  CALL G2CLPO (NCSW,NCSE,NCNE,NCNW,ICELL,IWARN)
                  IF (NDEDGE .EQ. 3) THEN
                     NVOID         = NVOID + 1
                     MRKDA2(NVOID) = ICELL
                  ENDIF
                ENDIF
C               %%%%   END_VERIFY_SUPERCELL
              ENDIF
C             ####   END_CHECK_FOR_COLLAPSE
            ENDIF
C           ****   END_CHECK_CELL_CENTER

190     CONTINUE
C
C       NOW CHECK THE NEIGHBOURS OF THE PREVIOUSLY COLLAPSED CELLS
C
200     KVOID = 0
        DO 210 JCELL = 1, NVOID

           ICELL = MRKDA2(JCELL)
           KS    = ICELG2(3,ICELL)
           KE    = ICELG2(5,ICELL)
           KN    = ICELG2(7,ICELL)
           KW    = ICELG2(9,ICELL)

C          CHECK IF SOUTHERN EDGE NODE EXISTS
C          ****   SOUTHERN_EDGE
           IF (KS .NE. 0) THEN
              ISONJ = NEIBG2(1,KS)
              IPAPJ = ICELG2(10,ISONJ)
              KCP   = ICELG2(1,IPAPJ)
              KSP   = ICELG2(3,IPAPJ)
              KEP   = ICELG2(5,IPAPJ)
              KNP   = ICELG2(7,IPAPJ)
              KWP   = ICELG2(9,IPAPJ)

              NBSWH  = NEIBG2(1,KSP)
              NBSEH  = NEIBG2(2,KSP)
              NBSEV  = NEIBG2(2,KEP)
              NBNEV  = NEIBG2(3,KEP)
              NBNEH  = NEIBG2(3,KNP)
              NBNWH  = NEIBG2(4,KNP)
              NBNWV  = NEIBG2(4,KWP)
              NBSWV  = NEIBG2(1,KWP)
              NDEDGE = 0
C
              IF (NBSWH .EQ. NBSEH) NDEDGE = NDEDGE + 1
              IF (NBSEV .EQ. NBNEV) NDEDGE = NDEDGE + 1
              IF (NBNEH .EQ. NBNWH) NDEDGE = NDEDGE + 1
              IF (NBSWV .EQ. NBNWV) NDEDGE = NDEDGE + 1
C
C             ####   CHECK_FOR_COLLAPSE
              IF (NDEDGE .GE. 3) THEN
```

```
                NCSW = NEIBG2(1,KCP)
                NCSE = NEIBG2(2,KCP)
                NCNE = NEIBG2(3,KCP)
                NCNW = NEIBG2(4,KCP)
                ISUP = ICELG2(10,NCSW)
C
C               %%%%   VERIFY_SUPERCELL
                IF (ISUP .EQ. IPAPJ) THEN
                  CALL G2CLPO (NCSW,NCSE,NCNE,NCNW,IPAPJ,IWARN)
                  IF (NDEDGE .EQ. 3) THEN
                     KVOID         = KVOID + 1
                     MRKCA2(NVOID) = IPAPJ
                  ENDIF
                ENDIF
C               %%%%   END_VERIFY_SUPERCELL
              ENDIF
C             ####   END_CHECK_FOR_COLLAPSE
            ENDIF
C           ****   SOUTHERN_EDGE

C           CHECK IF EASTERN EDGE NODE EXISTS
C           ****   EASTERN_EDGE
            IF (KE .NE. 0) THEN
               ISONJ = NEIBG2(2,KE)
               IPAPJ = ICELG2(10,ISONJ)
               KCP   = ICELG2(1,IPAPJ)
               KSP   = ICELG2(3,IPAPJ)
               KEP   = ICELG2(5,IPAPJ)
               KNP   = ICELG2(7,IPAPJ)
               KWP   = ICELG2(9,IPAPJ)

               NBSWH  = NEIBG2(1,KSP)
               NBSEH  = NEIBG2(2,KSP)
               NBSEV  = NEIBG2(2,KEP)
               NBNEV  = NEIBG2(3,KEP)
               NBNEH  = NEIBG2(3,KNP)
               NBNWH  = NEIBG2(4,KNP)
               NBNWV  = NEIBG2(4,KWP)
               NBSWV  = NEIBG2(1,KWP)
               NDEDGE = 0
C
               IF (NBSWH .EQ. NBSEH) NDEDGE = NDEDGE + 1
               IF (NBSEV .EQ. NBNEV) NDEDGE = NDEDGE + 1
               IF (NBNEH .EQ. NBNWH) NDEDGE = NDEDGE + 1
               IF (NBSWV .EQ. NBNWV) NDEDGE = NDEDGE + 1
C
C              ####   CHECK_FOR_COLLAPSE
               IF (NDEDGE .GE. 3) THEN
                 NCSW = NEIBG2(1,KCP)
                 NCSE = NEIBG2(2,KCP)
                 NCNE = NEIBG2(3,KCP)
                 NCNW = NEIBG2(4,KCP)
                 ISUP = ICELG2(10,NCSW)
C
C                %%%%   VERIFY_SUPERCELL
                 IF (ISUP .EQ. IPAPJ) THEN
                   CALL G2CLPO (NCSW,NCSE,NCNE,NCNW,IPAPJ,IWARN)
```

```
            IF (NDEDGE .EQ. 3) THEN
               KVOID         = KVOID + 1
               MRKCA2(NVOID) = IPAPJ
            ENDIF
          ENDIF
C         %%%%   END_VERIFY_SUPERCELL
        ENDIF
C       ####   END_CHECK_FOR_COLLAPSE
      ENDIF
C     ****   EASTERN_EDGE

C     CHECK IF NORTHERN EDGE NODE EXISTS
C     ****   NORTHERN_EDGE
      IF (KN .NE. 0) THEN
        ISONJ = NEIBG2(3,KN)
        IPAPJ = ICELG2(10,ISONJ)
        KCP   = ICELG2(1,IPAPJ)
        KSP   = ICELG2(3,IPAPJ)
        KEP   = ICELG2(5,IPAPJ)
        KNP   = ICELG2(7,IPAPJ)
        KWP   = ICELG2(9,IPAPJ)

        NBSWH  = NEIBG2(1,KSP)
        NBSEH  = NEIBG2(2,KSP)
        NBSEV  = NEIBG2(2,KEP)
        NBNEV  = NEIBG2(3,KEP)
        NBNEH  = NEIBG2(3,KNP)
        NBNWH  = NEIBG2(4,KNP)
        NBNWV  = NEIBG2(4,KWP)
        NBSWV  = NEIBG2(1,KWP)
        NDEDGE = 0
C
        IF (NBSWH .EQ. NBSEH) NDEDGE = NDEDGE + 1
        IF (NBSEV .EQ. NBNEV) NDEDGE = NDEDGE + 1
        IF (NBNEH .EQ. NBNWH) NDEDGE = NDEDGE + 1
        IF (NBSWV .EQ. NBNWV) NDEDGE = NDEDGE + 1
C
C       ####   CHECK_FOR_COLLAPSE
        IF (NDEDGE .GE. 3) THEN
          NCSW = NEIBG2(1,KCP)
          NCSE = NEIBG2(2,KCP)
          NCNE = NEIBG2(3,KCP)
          NCNW = NEIBG2(4,KCP)
          ISUP = ICELG2(10,NCSW)
C
C         %%%%   VERIFY_SUPERCELL
          IF (ISUP .EQ. IPAPJ) THEN
            CALL G2CLPO (NCSW,NCSE,NCNE,NCNW,IPAPJ,IWARN)
            IF (NDEDGE .EQ. 3) THEN
               KVOID         = KVOID + 1
               MRKCA2(NVOID) = IPAPJ
            ENDIF
          ENDIF
C         %%%%   END_VERIFY_SUPERCELL
        ENDIF
C       ####   END_CHECK_FOR_COLLAPSE
      ENDIF
```

```
C          ****   NORTHERN_EDGE

C          CHECK IF WESTERN EDGE NODE EXISTS
C          ****   WESTERN_EDGE
           IF (KW .NE. 0) THEN
              ISONJ = NEIBG2(4,KW)
              IPAPJ = ICELG2(10,ISONJ)
              KCP   = ICELG2(1,IPAPJ)
              KSP   = ICELG2(3,IPAPJ)
              KEP   = ICELG2(5,IPAPJ)
              KNP   = ICELG2(7,IPAPJ)
              KWP   = ICELG2(9,IPAPJ)

              NBSWH  = NEIBG2(1,KSP)
              NBSEH  = NEIBG2(2,KSP)
              NBSEV  = NEIBG2(2,KEP)
              NBNEV  = NEIBG2(3,KEP)
              NBNEH  = NEIBG2(3,KNP)
              NBNWH  = NEIBG2(4,KNP)
              NBNWV  = NEIBG2(4,KWP)
              NBSWV  = NEIBG2(1,KWP)
              NDEDGE = 0
C
              IF (NBSWH .EQ. NBSEH) NDEDGE = NDEDGE + 1
              IF (NBSEV .EQ. NBNEV) NDEDGE = NDEDGE + 1
              IF (NBNEH .EQ. NBNWH) NDEDGE = NDEDGE + 1
              IF (NBSWV .EQ. NBNWV) NDEDGE = NDEDGE + 1
C
C             ####   CHECK_FOR_COLLAPSE
              IF (NDEDGE .GE. 3) THEN
                NCSW = NEIBG2(1,KCP)
                NCSE = NEIBG2(2,KCP)
                NCNE = NEIBG2(3,KCP)
                NCNW = NEIBG2(4,KCP)
                ISUP = ICELG2(10,NCSW)
C
C               %%%%   VERIFY_SUPERCELL
                IF (ISUP .EQ. IPAPJ) THEN
                  CALL G2CLPO (NCSW,NCSE,NCNE,NCNW,IPAPJ,IWARN)
                  IF (NDEDGE .EQ. 3) THEN
                     KVOID         = KVOID + 1
                     MRKCA2(NVOID) = IPAPJ
                  ENDIF
                ENDIF
C               %%%%   END_VERIFY_SUPERCELL
              ENDIF
C             ####   END_CHECK_FOR_COLLAPSE
           ENDIF
C          ****   NORTHERN_EDGE

210     CONTINUE

C       UPDATE THE PREVIOUS LIST OF COLLAPSE CELLS
C
        NVOID = KVOID
        DO 220 JCELL = 1, NVOID
           MRKDA2(JCELL) = MRKCA2(JCELL)
```

```
220     CONTINUE

        NTIME = NTIME + 1
        IF (NTIME .GT. 20) GOTO 180
        IF (NVOID .NE.  0) GOTO 200

180     CONTINUE
        CALL G2NODE

        NVOID  = 0
        NTIME  = 1
C
        DO 60 ICELL = 1, NCELG2

C          ####   CHECK_ONLY_DIVIDED_CELLS

           IF (ICELG2(1,ICELL) .EQ. 0) THEN
              NDEDGE = 0
              DO 10 IEDGE = 3, 9, 2
                 IF (ICELG2(IEDGE,ICELL) .NE. 0) NDEDGE = NDEDGE + 1
10            CONTINUE
C
              IF (NDEDGE .EQ. 2) THEN
                IF (IAND(KAUXG2(ICELL),KLOOOF) .NE. 0) THEN
                   NVOID         = NVOID + 1
                   MRKDA2(NVOID) = ICELL
                   CALL G2DIVO (ICELL,IWARN)
                   GOTO 60
                ENDIF
C               ****   CHECK_FIRST_TYPE_VOID_CELLS
              ELSE IF (NDEDGE .GE. 3) THEN
                IF (NDEDGE .EQ. 3) THEN
                   NVOID         = NVOID + 1
                   MRKDA2(NVOID) = ICELL
                ENDIF
                CALL G2DIVO (ICELL,IWARN)
              ENDIF
C             ****   END CHECK_FIRST_TYPE_VOID_CELLS
           ENDIF
C          ####   END CHECK_ONLY_DIVIDED_CELLS

60      CONTINUE
C
C       NOW CHECK THE NEIGHBOURS OF THE PREVIOUSLY DIVIDED CELLS
C
80      KVOID = 0

        DO 150 JCELL = 1, NVOID

C          FIND THE ACTUAL CELL, ITS NODES AND NEIGHBOUR CELLS

           ICELL = MRKDA2(JCELL)
           KSW   = ICELG2(2,ICELL)
           KSE   = ICELG2(4,ICELL)
           KNE   = ICELG2(6,ICELL)
           KNW   = ICELG2(8,ICELL)
C
```

```
            NBSWV = NEIBG2(4,KSW)
            NBSWH = NEIBG2(2,KSW)
            NBSEH = NEIBG2(1,KSE)
            NBSEV = NEIBG2(3,KSE)
            NBNEV = NEIBG2(2,KNE)
            NBNEH = NEIBG2(4,KNE)
            NBNWH = NEIBG2(3,KNW)
            NBNWV = NEIBG2(1,KNW)
C
C           NCHECK INDICATES THE NUMBER OF CELLS TO BE CHECKED;
C           INB() HOLDS THESE CELLS (MAX 8)
C
            NCHECK = 0
C
C           SOUTHERN EDGE
C
            IF (NBSWH .NE. NBSEH) THEN
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBSWH
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBSEH
            ELSE IF (NBSWH .NE. 0) THEN
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBSWH
            ENDIF
C
C           EASTERN EDGE
C
            IF (NBSEV .NE. NBNEV) THEN
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBSEV
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBNEV
            ELSE IF (NBSEV .NE. 0) THEN
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBSEV
            ENDIF
C
C           NORTHERN EDGE
C
            IF (NBNEH .NE. NBNWH) THEN
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBNEH
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBNWH
            ELSE IF (NBNEH .NE. 0) THEN
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBNEH
            ENDIF
C
C           WESTERN EDGE
C
            IF (NBNWV .NE. NBSWV) THEN
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBNWV
               NCHECK      = NCHECK + 1
               INB(NCHECK) = NBSWV
```

```
            ELSE IF (NBNWV .NE. 0) THEN
               NCHECK      = NCHECK + 1
             - INB(NCHECK) = NBNWV
            ENDIF
C
C           NOW CHECK ALL THE PREVIOUSLY COLLECTED NEIGHBOUR CELLS
C
            DO 140 KCELL = 1, NCHECK
               LCELL = INB(KCELL)
C              ####   CHECK_ONLY_DIVIDED_CELLS
               IF (ICELG2(1,LCELL) .EQ. 0) THEN
                  NDEDGE = 0
                  DO 90 IEDGE = 3, 9, 2
                     IF (ICELG2(IEDGE,LCELL) .NE. 0) NDEDGE = NDEDGE + 1
90                CONTINUE
C
                  IF (NDEDGE .EQ. 2) THEN
                    IF (IAND(KAUXG2(LCELL),KLOOOF) .NE. 0) THEN
                       KVOID         = KVOID + 1
                       MRKCA2(KVOID) = LCELL
                       CALL G2DIVO (LCELL,IWARN)
                       GOTO 140
                    ENDIF
C                   ****   CHECK_FIRST_TYPE_VOID_CELLS
                    ELSE IF (NDEDGE .GE. 3) THEN
                      IF (NDEDGE .EQ. 3) THEN
                         KVOID         = KVOID + 1
                         MRKCA2(KVOID) = LCELL
                      ENDIF
                      CALL G2DIVO (LCELL,IWARN)
                    ENDIF
C                   ****   END CHECK_FIRST_TYPE_VOID_CELLS
               ENDIF
C              ####   END CHECK_ONLY_DIVIDED_CELLS
140         CONTINUE

150      CONTINUE

C        UPDATE THE PREVIOUS LIST OF CELLS
C
         NVOID = KVOID
         DO 160 JCELL = 1, NVOID
            MRKDA2(JCELL) = MRKCA2(JCELL)
160      CONTINUE
C
         NTIME = NTIME + 1
         IF (NTIME .GT. 20) RETURN
         IF (NVOID .NE.  0) GOTO 80

         RETURN
         END
```

A2VOID

```
      SUBROUTINE A2VOID

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'

      DIMENSION INB(12)
      LOGICAL   IWRITE

C*********************************************************************

C     THIS SUBROUTINE DETECTS THE VOID CELLS AFTER THE PREVIOUS
C     ADAPTATION CYCLE OF GRID DIVISION, EXTENSION AND COLLAPSE.
C     A VOIDS CELL IS THE ONE WITH ONE OF THE FOLLOWING PROPERTIES:
C       1. HAS ATLEAST THREE DIVIDED EDGES.
C       2. HAS TWO DIVIDED EDGES ON A BOUNDARY
C       3. HAS TWO DIVIDED EDGES AND IS CONTIGUOUS TO A SIMILAR CELL
C     NOTE THAT THE ISLANDS ARE TOLERATED BUT THE VOIDS AREN'T.
C     ONLY THE CEWIC CELLS NEED BE CHECKED FOR VOIDS AND ISLANDS.
C
C     NVOID : THE NUMBER OF CELLS WHICH ARE DETECTED TO BE VOID
C             CELLS (STORED IN MRKDA2 IN PASS 1 AND IN MRKCA2 IN
C             PASS 2)
C     NEDG2 : THE NUMBER OF CELLS WITH TWO DIVIDED EDGES WHICH ARE
C             ALSO NOT ON THE BOUNDARY (STORED IN WORKA2)

C*********************************************************************
C
C     WANT DEBUG PRINT ?

      IWRITE = IDBGA2 .EQ. 12 .OR. IDBGA2 .GT. 1000
C
C      + + + + + + + + + + + + + + + + +
C      + NBNWC | NBNWH + NBNEH | NBNEC +
C      +       |       +       |       +
C      +-------+-------+-------+-------+
C      +       |KNW   KN    KNE|       +
C      + NBNWV |               | NBNEV +
C      + + + + +KW   ICELL   KE+ + + + +
C      + NBSWV |               | NBSEV +
C      +       |KSW   KS    KSE|       +
C      +-------+-------+-------+-------+
C      +       |       +       |       +
C      + NBSWC | NBSWH + NBSEH | NBSEC +
C      + + + + + + + + + + + + + + + + +
C
C     INITIALIZE THE NUMBER OF VOID CELLS, NDEDGE INDICATES THE
C     NUMBER OF DIVIDED EDGES

      NVOID = 0
      NEDG2 = 0
```

```
      NTIME = 1
C
      DO 70 ICELL = 1, NCELG2

C        ####   CHECK_ONLY_DIVIDED_CELLS
C
         IF (ICELG2(1,ICELL) .EQ. 0) THEN
            NDEDGE = 0
            DO 10 IEDGE = 3, 9, 2
               IF (ICELG2(IEDGE,ICELL) .NE. 0) NDEDGE = NDEDGE + 1
10          CONTINUE
C
C           ****   CHECK_THIRD_TYPE_VOID_CELLS
            IF (NDEDGE .EQ. 2) THEN
              IF (IAND(KAUXG2(ICELL),KLOOOF) .NE. 0) GOTO 60
              KSW   = ICELG2(2,ICELL)
              KSE   = ICELG2(4,ICELL)
              KNE   = ICELG2(6,ICELL)
              KNW   = ICELG2(8,ICELL)
              NBSWV = NEIBG2(4,KSW)
              NBSWH = NEIBG2(2,KSW)
              NBSEH = NEIBG2(1,KSE)
              NBSEV = NEIBG2(3,KSE)
              NBNEV = NEIBG2(2,KNE)
              NBNEH = NEIBG2(4,KNE)
              NBNWH = NEIBG2(3,KNW)
              NBNWV = NEIBG2(1,KNW)
C
C             JCHECK INDICATES THAT THE THIRD TYPE VOID CELLS EXIST
              JCHECK = 0

C             SOUTHERN EDGE
              IF (NBSWH .EQ. NBSEH) THEN
                 NDEDGE = 0
                 DO 20 IEDGE = 3, 9, 2
                   IF (ICELG2(IEDGE,NBSWH) .NE. 0) NDEDGE =NDEDGE+1
20               CONTINUE
                 IF (NDEDGE .EQ. 2) JCHECK = 1
              ENDIF

C             EASTERN EDGE
              IF (NBSEV .EQ. NBNEV) THEN
                 NDEDGE = 0
                 DO 30 IEDGE = 3, 9, 2
                   IF (ICELG2(IEDGE,NBSEV) .NE. 0) NDEDGE =NDEDGE+1
30               CONTINUE
                 IF (NDEDGE .EQ. 2) JCHECK = 1
              ENDIF

C             NORTHERN EDGE
              IF (NBNEH .EQ. NBNWH) THEN
                 NDEDGE = 0
                 DO 40 IEDGE = 3, 9, 2
                   IF (ICELG2(IEDGE,NBNWH) .NE. 0) NDEDGE =NDEDGE+1
40               CONTINUE
                 IF (NDEDGE .EQ. 2) JCHECK = 1
              ENDIF
```

```
C              WESTERN EDGE
               IF (NBNWV .EQ. NBSWV) THEN
                  NDEDGE = 0
                  DO 50 IEDGE = 3, 9, 2
                    IF (ICELG2(IEDGE,NBSWV) .NE. 0) NDEDGE =NDEDGE+1
50                CONTINUE
                  IF (NDEDGE .EQ. 2) JCHECK = 1
               ENDIF

               IF (JCHECK .NE. 0) THEN
                  NEDG2         = NEDG2 + 1
                  WORKA2(NEDG2) = ICELL
               ENDIF
C             ****   END CHECK_THIRD_TYPE_VOID_CELLS

C             ****   CHECK_FIRST_TYPE_VOID_CELLS
              ELSE IF (NDEDGE .GE. 3) THEN

60              NVOID         = NVOID + 1
                MRKDA2(NVOID) = ICELL
                IWARN         = 0
                CALL G2DIV0 (ICELL,IWARN)
                WRITE(6,*) ' A2VOID: DIVIDED CELL IS ',ICELL
                IF (IWARN .NE. 0) WRITE(JTERM0,1000) IWARN, ICELL
C               SEE IF DEBUG CHECK IS NEEDED
                IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
                   NERR = 0
                   CALL CHKBN2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
                   CALL CHKNC2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
                   CALL CHKNN2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
                   CALL CHKSP2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
                ENDIF
              ENDIF
C             ****   END CHECK_FIRST_TYPE_VOID_CELLS
            ENDIF
C           ####   END CHECK_ONLY_DIVIDED_CELLS

70        CONTINUE

C         RESET THE NUMBER OF THIRD TYPE CELLS, ADJUST THE VOID CELL
C         ARRAY AND DIVIDE THESE CELLS
C
          DO 80 JCELL = 1, NEDG2
             ICELL         = NINT(WORKA2(JCELL))
             IF (ICELG2(1,ICELL) .EQ. 0) THEN
                NVOID         = NVOID + 1
                MRKDA2(NVOID) = ICELL
                IWARN         = 0
                CALL G2DIV0 (ICELL,IWARN)
                WRITE(6,*) ' A2VOID: DIVIDED CELL IS ',ICELL
                IF (IWARN .NE. 0) WRITE(JTERM0,1000) IWARN, ICELL
C               SEE IF DEBUG CHECK IS NEEDED
                IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
                   NERR = 0
                   CALL CHKBN2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
                   CALL CHKNC2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
```

```
               CALL CHKNN2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
     -         CALL CHKSP2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
            ENDIF
          ENDIF
80      CONTINUE
        NEDG2 = 0
C
C       PRINT OUT PARAMETERS
C
        IF (IWRITE)  THEN
          WRITE(JDEBUG,1100)
          WRITE(JDEBUG,1200)
          WRITE(JDEBUG,1300)
          WRITE(JDEBUG,1400) NVOID, NTIME
          WRITE(JDEBUG,1500) (MRKDA2(I), I = 1, NVOID)
        ENDIF
C
C       NOW CHECK THE NEIGHBOURS OF THE PREVIOUSLY DIVIDED CELLS
C
90      KVOID = 0

        DO 170 JCELL = 1, NVOID

C         FIND THE ACTUAL CELL, ITS NODES AND NEIGHBOUR CELLS

          ICELL = MRKDA2(JCELL)
          KSW   = ICELG2(2,ICELL)
          KSE   = ICELG2(4,ICELL)
          KNE   = ICELG2(6,ICELL)
          KNW   = ICELG2(8,ICELL)
C
          NBSWV = NEIBG2(4,KSW)
          NBSWC = NEIBG2(1,KSW)
          NBSWH = NEIBG2(2,KSW)
          NBSEH = NEIBG2(1,KSE)
          NBSEC = NEIBG2(2,KSE)
          NBSEV = NEIBG2(3,KSE)
          NBNEV = NEIBG2(2,KNE)
          NBNEC = NEIBG2(3,KNE)
          NBNEH = NEIBG2(4,KNE)
          NBNWH = NEIBG2(3,KNW)
          NBNWC = NEIBG2(4,KNW)
          NBNWV = NEIBG2(1,KNW)

C         NCHECK INDICATES THE NUMBER OF CELLS TO BE CHECKED;
C         INB() HOLDS THESE CELLS (MAX 12)
C         ******* ARE THE CORNER CELLS REALLY NECESSARY ? *******

          NCHECK = 0
C
C         SOUTHWEST CORNER
C
          IF (NBSWC .NE. 0) THEN
            NCHECK      = NCHECK + 1
            INB(NCHECK) = NBSWC
          ENDIF
C
```

```
C          SOUTHERN EDGE
C
           IF (NBSWH .NE. NBSEH) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBSWH
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBSEH
           ELSE IF (NBSWH .NE. 0) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBSWH
           ENDIF
C
C          SOUTHEAST CORNER
C
           IF (NBSEC .NE. 0) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBSEC
           ENDIF
C
C          EASTERN EDGE
C
           IF (NBSEV .NE. NBNEV) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBSEV
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNEV
           ELSE IF (NBSEV .NE. 0) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBSEV
           ENDIF
C
C          NORTHEAST CORNER
C
           IF (NBNEC .NE. 0) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNEC
           ENDIF
C
C          NORTHERN EDGE
C
           IF (NBNEH .NE. NBNWH) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNEH
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNWH
           ELSE IF (NBNEH .NE. 0) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNEH
           ENDIF
C
C          NORTHWEST CORNER
C
           IF (NBNWC .NE. 0) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNWC
           ENDIF
C
```

```
C          WESTERN EDGE
C
           IF (NBNWV .NE. NBSWV) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNWV
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBSWV
           ELSE IF (NBNWV .NE. 0) THEN
              NCHECK      = NCHECK + 1
              INB(NCHECK) = NBNWV
           ENDIF
C
C          NOW CHECK ALL THE PREVIOUSLY COLLECTED NEIGHBOUR CELLS
C
           DO 160 KCELL = 1, NCHECK
              LCELL = INB(KCELL)
C             ####   CHECK_ONLY_DIVIDED_CELLS
              IF (ICELG2(1,LCELL) .EQ. 0) THEN
                 NDEDGE = 0
                 DO 100 IEDGE = 3, 9, 2
                    IF (ICELG2(IEDGE,LCELL) .NE. 0) NDEDGE = NDEDGE + 1
100              CONTINUE
C
C                ****   CHECK_THIRD_TYPE_VOID_CELLS
                 IF (NDEDGE .EQ. 2) THEN
                   IF (IAND(KAUXG2(LCELL),KL000F) .NE. 0) GOTO 150
                   JCHECK = 0

C                  SOUTHERN EDGE
                   IF (NBSWH .EQ. NBSEH) THEN
                     NDEDGE = 0
                     DO 110 IEDGE = 3, 9, 2
                       IF (ICELG2(IEDGE,NBSWH) .NE. 0) NDEDGE =NDEDGE+1
110                  CONTINUE
                     IF (NDEDGE .EQ. 2) JCHECK = 1
                   ENDIF

C                  EASTERN EDGE
                   IF (NBSEV .EQ. NBNEV) THEN
                     NDEDGE = 0
                     DO 120 IEDGE = 3, 9, 2
                       IF (ICELG2(IEDGE,NBSEV) .NE. 0) NDEDGE =NDEDGE+1
120                  CONTINUE
                     IF (NDEDGE .EQ. 2) JCHECK = 1
                   ENDIF

C                  NORTHERN EDGE
                   IF (NBNEH .EQ. NBNWH) THEN
                     NDEDGE = 0
                     DO 130 IEDGE = 3, 9, 2
                       IF (ICELG2(IEDGE,NBNWH) .NE. 0) NDEDGE =NDEDGE+1
130                  CONTINUE
                     IF (NDEDGE .EQ. 2) JCHECK = 1
                   ENDIF

C                  WESTERN EDGE
                   IF (NBNWV .EQ. NBSWV) THEN
```

```
                  NDEDGE = 0
                  DO 140 IEDGE = 3, 9, 2
     -              IF (ICELG2(IEDGE,NBSWV) .NE. 0) NDEDGE =NDEDGE+1
140               CONTINUE
                  IF (NDEDGE .EQ. 2) JCHECK = 1
                ENDIF

                IF (JCHECK .NE. 0) THEN
                  NEDG2         = NEDG2 + 1
                  WORKA2(NEDG2) = LCELL
                ENDIF
C             ****   END CHECK_THIRD_TYPE_VOID_CELLS
C             ****   CHECK_FIRST_TYPE_VOID_CELLS
              ELSE IF (NDEDGE .GE. 3) THEN

150             KVOID         = KVOID + 1
                MRKCA2(KVOID) = LCELL
                IWARN         = 0
                CALL G2DIVO (LCELL,IWARN)
                WRITE(6,*) ' A2VOID: DIVIDED CELL IS ',LCELL
                IF (IWARN .NE. 0) WRITE(JTERMO,1000) IWARN, LCELL
C               SEE IF DEBUG CHECK IS NEEDED
                IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
                   NERR = 0
                   CALL CHKBN2 (LCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                   CALL CHKNC2 (LCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                   CALL CHKNN2 (LCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                   CALL CHKSP2 (LCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                ENDIF           ! DEBUG CHECK
              ENDIF
C             ****   END CHECK_FIRST_TYPE_VOID_CELLS
            ENDIF
C           ####   END CHECK_ONLY_DIVIDED_CELLS
160       CONTINUE
C
170     CONTINUE

C       UPDATE THE PREVIOUS LIST OF CELLS
C
        NVOID = KVOID
        DO 180 JCELL = 1, NVOID
           MRKDA2(JCELL) = MRKCA2(JCELL)
180     CONTINUE
C
C       RESET THE NUMBER OF THIRD TYPE CELLS, ADJUST THE VOID CELL
C       ARRAY AND DIVIDE THESE CELLS
C
        DO 190 JCELL = 1, NEDG2
           ICELL         = NINT(WORKA2(JCELL))
           IF (ICELG2(1,ICELL) .EQ. 0) THEN
              NVOID          = NVOID + 1
              MRKDA2(NVOID) = ICELL
              IWARN          = 0
              CALL G2DIVO (ICELL,IWARN)
              WRITE(6,*) ' A2VOID: DIVIDED CELL IS ',ICELL
              IF (IWARN .NE. 0) WRITE(JTERMO,1000) IWARN, ICELL
C             SEE IF DEBUG CHECK IS NEEDED
```

```
            IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
               NERR = 0
               CALL CHKBN2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
               CALL CHKNC2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
               CALL CHKNN2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
               CALL CHKSP2 (ICELL, 0, 0, 0, 0, NERR,'AFTDIV')
            ENDIF
         ENDIF
190     CONTINUE
        NEDG2 = 0
C
C       PRINT OUT PARAMETERS
C
        NTIME = NTIME + 1
        IF (IWRITE)  THEN
           WRITE(JDEBUG,1400) NVOID, NTIME
           WRITE(JDEBUG,1500) (MRKDA2(I), I = 1, NVOID)
        ENDIF
C
        IF (NTIME .GT. 20) GOTO 200
        IF (NVOID .NE.  0) GOTO 90
C
C       CHECK FOR ISLANDS
C
200     DO 210 ICELL = 1, NCELG2

           IF (ICELG2(1,ICELL) .NE. 0) THEN
              NDEDGE = 0
              KC     = ICELG2(1,ICELL)
              KS     = ICELG2(3,ICELL)
              KE     = ICELG2(5,ICELL)
              KN     = ICELG2(7,ICELL)
              KW     = ICELG2(9,ICELL)
              NBSWH  = NEIBG2(1,KS )
              NBSEH  = NEIBG2(2,KS )
              NBSEV  = NEIBG2(2,KE )
              NBNEV  = NEIBG2(3,KE )
              NBNEH  = NEIBG2(3,KN )
              NBNWH  = NEIBG2(4,KN )
              NBNWV  = NEIBG2(4,KW )
              NBSWV  = NEIBG2(1,KW )
C
              IF (NBSWH .EQ. NBSEH) NDEDGE = NDEDGE + 1
              IF (NBSEV .EQ. NBNEV) NDEDGE = NDEDGE + 1
              IF (NBNEH .EQ. NBNWH) NDEDGE = NDEDGE + 1
              IF (NBSWV .EQ. NBNWV) NDEDGE = NDEDGE + 1
C
              IF (NDEDGE .EQ. 4) THEN
                NCSW = NEIBG2(1,KC)
                NCSE = NEIBG2(2,KC)
                NCNE = NEIBG2(3,KC)
                NCNW = NEIBG2(4,KC)
                ISUP = ICELG2(10,NCSW)
C
                IF (ISUP .EQ. ICELL)
     1            CALL G2CLPO (NCSW,NCSE,NCNE,NCNW,ICELL,IWARN)
```

```
            ENDIF
          ENDIF
C
210     CONTINUE
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(5X,'WARNING #',I3,2X,'ISSUED FOR CELL',I5)
1100    FORMAT(//10X,'----------------------' )
1200    FORMAT(  10X,'DEBUG PRINT FROM A2VOID' )
1300    FORMAT(  10X,'----------------------'/)
1400    FORMAT(/10X,'NUMBER OF VOID CELLS = 'I5,2X,'AFTER PASS',I5/
     1          10X,'THE VOID CELLS ARE :'/)
1500    FORMAT(20I5)

        RETURN
        END
```

C2CHEK

```
        PROGRAM C2CHEK
C
        PARAMETER (MKOUNT=100)
        INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] CHCOMN.INC/LIST'
C
        DIMENSION TEMPER(3), RHS(3), AMAT(3,3), SOLN(3)
        DIMENSION TEMP$(MKOUNT), AKEQ$(MKOUNT)
C
C***********************************************************************
C
C       THIS PROGRAM CHECKS THE VALIDITY OF THE EQUILIBRIUM RATE CONSTANT
C       ARHENIUS MODEL
C
C***********************************************************************
C
C       SETUP INPUT/OUTPUT UNITS
C
        JTERMI = 5
        JTERMO = 6
        JPRINT = 7
        JNCHEK = 9
C
        OPEN (UNIT=JPRINT, FILE='OUCHEK.DAT', STATUS='NEW')
        OPEN (UNIT=JNCHEK, FILE='INCHEK.DAT', STATUS='OLD')
C
C       READ THE APPROPRITE INFORMATION FORM JNCHEK

        READ  (JNCHEK,*) TINIT, TINCR, TFINAL, TREFCH
        READ  (JNCHEK,*) TEMPER(1), TEMPER(2), TEMPER(3)
        READ  (JNCHEK,*) NSPECH
```

```
C       READ THE REACTION NUMBER AND REACTION COEFFICIENTS, FOR BOTH
C       REACTANT AND PRODUCT SIDES FOR EACH REACTION

        READ(JNCHEK,*) (IALPCH(IS,1),IS=1,NSPECH)
        READ(JNCHEK,*) (IBETCH(IS,1),IS=1,NSPECH)
        DO 10 IS = 1, NSPECH
           BMIACH(IS,1) = IBETCH(IS,1) - IALPCH(IS,1)
10      CONTINUE
C
C       READ THE SPECIES NUMBER IS, ATOMIC WEIGHT, SPECIFIC HEATS
C       (CP IN KJ/KMOL/K), HEAT OF FORMATION (KJ/KMOL), AND
C       ENTROPY (KJ/KMOL/K) AT  THE REFERENCE CONDITIONS FOR EACH SPECIES
C       CONVERT SOME OF THESE TO /KG BASIS TO BE CONSISTENT WITH THE
C       STAR CODE
C
        DO 20 IS = 1, NSPECH
            READ(JNCHEK,*) ISP, AMWTCH(IS), SPCPCH(IS),
     1                     FMHTCH(IS), ENTRCH(IS), SPBSCH(IS)
            SPCPCH(IS) = 1000.*SPCPCH(IS)/AMWTCH(IS)
            SPBSCH(IS) = 1000.*SPBSCH(IS)/AMWTCH(IS)
            FMHTCH(IS) = 1000.*FMHTCH(IS)/AMWTCH(IS)
            ENTRCH(IS) = 1000.*ENTRCH(IS)
20      CONTINUE
C
C       SEE IF OTHER REFERENCE VALUES ARE AVAILABLE FOR COMPARISION
C       PURPOSES FOR THE EQUILIBRIUM RATE CONSTANTS
C       IPOWER > 0 REFERENCE IS AVAILABLE
C                1 USE FULL REGRESSION MODEL
C                2 USE ENEREF FOR REGRESSION MODEL
C                4 USE ENEREF AND EXPREF FOR REGRESSION MODEL
C
        READ (JNCHEK,*) IPOWER
        IF (IPOWER .GT. 0) THEN
           READ (JNCHEK,*) PREREF, EXPREF, ENEREF
        ENDIF
C
        IF (IPOWER .LE. 1) THEN
           DO 30 IT = 1, 3
              TEMP = TEMPER(IT)
              AT   = LOG(TEMP)
              RT   = 1./TEMP
C             GET THE LOG OF EQUILIBRIUM CONSTANT FOR REACTION 1
              CALL CHKCRE (TEMP, AKEQ)
              RHS(IT) = AKEQ
              AMAT(IT,1) = 1.
              AMAT(IT,2) = AT
              AMAT(IT,3) = -RT
30         CONTINUE
           CALL GAUSS2(AMAT,RHS,SOLN,3,3)
           PREECH(1) = SOLN(1)
           EXPECH(1) = SOLN(2)
           ENEECH(1) = SOLN(3)
        ELSE IF (IPOWER .EQ. 2) THEN
           ENEECH(1) = ENEREF
           DO 40 IT = 1, 3, 2
              TEMP = TEMPER(IT)
```

```
            AT   = LOG(TEMP)
            RT   = 1./TEMP
C          - GET THE LOG OF EQUILIBRIUM CONSTANT FOR REACTION 1
            CALL CHKCRE (TEMP, AKEQ)
            RHS(IT) = AKEQ + ENEREF*RT
            AMAT(IT,1) = 1.
            AMAT(IT,2) = AT
40        CONTINUE
          CALL GAUSS2(AMAT,RHS,SOLN,2,3)
          PREECH(1) = SOLN(1)
          EXPECH(1) = SOLN(2)
        ELSE
          ENEECH(1) = ENEREF
          EXPECH(1) = EXPREF
        ENDIF
C
C       WRITE THE CONSTANTS FOR THE MODEL
C
        WRITE(JPRINT,50) PREECH(1), EXPECH(1), ENEECH(1)
50      FORMAT(5X,5G14.5)

        TEMP   = TINIT
        SUMKEQ = 0.
        NDATA  = 0

60      ALOGT = ALOG(TEMP)
        RTEMP = 1./TEMP
C
        CALL CHKCRE (TEMP,AKEQ)
C
        SUMKEQ       = SUMKEQ + AKEQ - EXPREF*ALOGT + ENEREF*RTEMP
        NDATA        = NDATA + 1
        TEMP$(NDATA) = TEMP
        AKEQ$(NDATA) = AKEQ
        TEMP         = TEMP + TINCR
        IF (NDATA .GE. MKOUNT) GOTO 70
        IF (TEMP  .LE. TFINAL) GOTO 60

70      IF (IPOWER .EQ. 4) THEN
          SUMKEQ = SUMKEQ/NDATA
          WRITE(JPRINT,*)
          WRITE(JPRINT,*) ' AVERAGE PRE-EXP FACTOR',SUMKEQ
          PREECH(1) = SUMKEQ
        ENDIF

        WRITE(38,*) NDATA
        WRITE(39,*) NDATA

        DO 80 IDATA = 1, NDATA
          TEMP  = TEMP$(IDATA)
          AKEQ  = AKEQ$(IDATA)
          ALOGT = ALOG(TEMP)
          RTEMP = 1./TEMP
C
          AKEMOD = PREECH(1) + EXPECH(1)*ALOGT - ENEECH(1)*RTEMP
          AKEREF = PREREF + EXPREF*ALOGT - ENEREF*RTEMP
```

```
          WRITE(JPRINT,50) TEMP, AKEQ, AKEMOD, AKEREF
          WRITE(38,*) TEMP, AKEQ
          WRITE(39,*) TEMP, AKEREF

80      CONTINUE

C
C       ---------------------------------
C       EXAMPLE OF INPUT FILE: INCHEK.DAT
C       ---------------------------------
C
C    1000,100,3000,298              TINIT,TINCR,TFINAL,TREFCH
C    1000,2000,3000                 TEMPER(1), TEMPER(2), TEMPER(3)
C    5                              TOTAL NUMBER OF SPECIES
C    0   2   1   0   0              REACTION COEFFICIENTS
C    0   0   0   2   0
C  1 31.999   30.559   0.        205.142   0.34485E-2  IS, AMWT,CP,HF,SO,BS
C  2 17.008   28.071   39463.    183.703   0.30943E-2
C  3  2.016   27.290   0.        130.684   0.33530E-2
C  4 18.015   32.469   -241827.  188.833   0.86358E-2
C  5 28.013   29.282   0.        191.611   0.30233E-2
C  1                              REFERENCE VALUES ARE AVAILABLE
C -19.7367   1.0      -69415      CF, ETA, E-TERM (THIS IS POSITIVE!)
C
        STOP
        END

        SUBROUTINE CHKCRE (TEMP, AKEQ)
C
        INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
        INCLUDE '[PERVAIZ.TWODO.INC] CHCOMN.INC/LIST'
C
        DIMENSION GIBBS(MSPECH)
C
C       GIBBS(S) IS THE SPECIFIC MOLAL GIBBS FUNCTION FOR SPECIES S
C       DELGIB IS THE TOTAL GIBBS FUNCTION CHANGE FOR A REACTION R
C       SUMCOF IS THE SUM OF THE COEFFIENTS (DELTA_N) USEFUL IN THE
C       COMPUTATION OF EQUILIBRIUM CONSTANT
C       AKPR IS THE EQUILIBRIUM CONSTANT BASED ON PARTIAL PRESSURES
C       AKEQ IS THE EQUILIBRIUM CONSTANT BASED ON CONCENTRATIONS
C
C       UNIVERSAL GAS CONSTANT IN J/KMOL/K
        UGASFL = 8.31434E03
        PRESCH = 1.0125E5

        DO 10 IS = 1, NSPECH
          GIBBS(IS) = FMHTCH(IS)*AMWTCH(IS) - TEMP*ENTRCH(IS)
     1              + SPCPCH(IS)*AMWTCH(IS)*(TEMP-TREFCH)
     2              + 0.5*SPBSCH(IS)*AMWTCH(IS)*(TEMP**2-TREFCH**2)
     3              - TEMP*SPCPCH(IS)*AMWTCH(IS)*LOG(TEMP/TREFCH)
     4              - TEMP*SPBSCH(IS)*AMWTCH(IS)*(TEMP-TREFCH)
C
10      CONTINUE
C
C       COMPUTE THE (LOG OF) EQUILIBRIUM CONSTANTS FOR THE REACTION
C
        DELGIB = 0.
```

```
      SUMCOF = 0.
        DO 20 IS = 1, NSPECH
          DELGIB = DELGIB + BMIACH(IS,1)*GIBBS(IS)
          SUMCOF = SUMCOF - BMIACH(IS,1)
20      CONTINUE
      AKPR     = -DELGIB/TEMP/UGASFL
      AKEQ = AKPR + SUMCOF*LOG(UGASFL*TEMP/PRESCH)
C
      RETURN
      END
```

C2EQDI

```
      SUBROUTINE C2EQDI

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'

C***********************************************************************

C     THIS SUBROUTINE COMPUTES THE EQUILIBRIUM CONCENTRATION FOR THE
C     LIGHT-HILL DISSOCIATING GAS MODEL.

C***********************************************************************

      IF (KROGER .NE. 2) RETURN
      CONSTN = 0.5*AMWTCH(1)*EXP(PREECH(1))
      ETA    = EXPECH(1)
      ENERGY =-ENEECH(1)

      DO 100 INODE = 1, NNODG2
        RHO             = DPENG2(1,INODE)*RHORFL
        TEMP            = TEMPG2(INODE)*TREFFL
        TETA            = TEMP**ETA
        EXTERM          = EXP(ENERGY/TEMP)
        BETA            = CONSTN*TETA*EXTERM/RHO
        DISCRI          = BETA*(BETA + 4.)
        ALPHAN          = 0.5*(SQRT(DISCRI) - BETA)
        ALPHAO          = DPENG2(5,INODE)/DPENG2(1,INODE)
        ALPHA           = 0.1*(9.*ALPHAO+ALPHAN)
        DPENG2(5,INODE) = ALPHA*DPENG2(1,INODE)
100   CONTINUE
      RETURN
      END
```

C2EQRC

```
      SUBROUTINE C2EQRC

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE USES THE FIRST REACTION OF THE ROGERS AND CHINITZ
C     MODEL AS EQUILIBRIUM REACTION.
C     THE SUBROUTINE ALSO RECOMPUTES THE PRESSURES AND TEMPERATURES
C
C*********************************************************************
C
C
      IF (KROGER .NE. 1    ) RETURN
      IF (TRIGCH .GT. 1500.) RETURN
C
C     SCAN ALL THE INTERIOR NODES FOR THE ROGERS AND CHINITZ MODEL
C     WHERE THE TEMPERATURE EXCEEDS A SPECIFIED VALUE
C
      DO 50 INODE = 1, NNODG2
C
         TEMPD = TREFFL*TEMPG2(INODE)
C
         IF (TEMPD .LT. TRIGCH) GOTO 50
C
C        SKIP BOUNDARY NODES
         DO 10 IBND = 1, NBNDG2
            IF (IBNDG2(1,IBND) .EQ. INODE) THEN
               IF (IBNDG2(5,IBND) .NE. 2) GOTO 20
               GOTO 50
            ENDIF
10       CONTINUE
C
20       RHORPR = DPENG2(1,INODE)
         YO2OLD = DPENG2(5,INODE)/RHORPR
         YOHOLD = DPENG2(6,INODE)/RHORPR
         YH2OLD = DPENG2(7,INODE)/RHORPR
         AKEQ   = 117.31948*EXP(-8992/TEMPD)
         YOHNEW = SQRT(YH2OLD*YO2OLD*AKEQ)
         DELTAY = 0.5*(YOHNEW-YOHOLD)/AMWTCH(2)
         YO2NEW = YO2OLD - AMWTCH(1)*DELTAY
         YH2NEW = YH2OLD - AMWTCH(3)*DELTAY
C
C        RESET THE DEPENDENT VARIABLES
C
         DPENG2(5,INODE) = RHORPR*YO2NEW
         DPENG2(6,INODE) = RHORPR*YOHNEW
         DPENG2(7,INODE) = RHORPR*YH2NEW
C
C        DETERMINE THE PRESSURE AND TEMPERATURE
```

```
C
         CALL E2PRMT(INODE,1)
50      CONTINUE

        RETURN
        END
```

C2HELP

```
        SUBROUTINE C2HELP
C
        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] HEXCOD.INC     '
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        CHARACTER  CHARBG*120
C
C
C***********************************************************************
C
C       THIS SUBROUTINE HELPS IN GENERATING THE CONSTANTS FOR CHEMISTRY
C       TO BE READ BY THE UNIT JREADC IN C2INIT.  THIS SUBROUTINE IS
C       NEEDED BECAUSE THE INPUT FILE BECOMES COMPLICATED DUE TO A
C       LARGE NUMBER OF TRANSFER STATEMENTS IN C2INIT.
C
C***********************************************************************
C
        JCHELP = 51

C       KROGER DENOTES THE SPECIAL TYPE OF REACTION MODEL TO BE USED.
C           KROGER = 1   : USE ROGER AND CHINITZ MODEL
C           KROGER = 2   : USE LIGHT HILL SINGLE DISSOCIATING GAS MODEL
C           KROGER = 3   : USE A SINGLE NON-REACTING GAS
C
C       SET UP THE CONSTANTS FOR ROGER AND CHINITZ MODEL, FOR THIS
C       MODEL THE SPECIES MUST BE ORDERED AS O2, H2O, H2, OH AND N2
C
        IF (KROGER .EQ. 1) THEN
          NSPECH = 5
          NREACH = 2
          NINRCH = 1
        ENDIF
C
C       FOR THE LIGHT HILL GAS MODEL THE SPECIES ARE ORDERED A, A2
C
        IF (KROGER .EQ. 2) THEN
          NSPECH = 2
          NREACH = 1
          NINRCH = 0
        ENDIF
C
        IF (KROGER .EQ. 3) THEN
          NSPECH = 1
```

```
          NREACH = 0
          NINRCH = 0
        ENDIF
C
C       WRITE PARAMETER FOR DEBUG PRINTING
C
        CHARBG        = ' '
        CHARBG( 1: 6) = 'IDBGCH'
        CHARBG(75:99) = '! DEBUG PARAMETER'

        WRITE(JCHELP,1000) CHARBG(1:99)
C
C       WRITE THE INITIAL MASS FRACTIONS YSPECH(S) FOR ALL SPECIES S

        CHARBG        = ' '
        CHARBG(71:99) = '! MASS FRACTION  '
        DO 10 IS = 1, NSPECH
           CHARBG( 1:10) = 'YSPECH(  )'
           CHARBG(21:30) = 'YSPECH(  )'
           WRITE (CHARBG( 8: 9), 1100) IS
           WRITE (CHARBG(28:29), 1100) IS
           WRITE(JCHELP,1200) IS,CHARBG(1:99)
10      CONTINUE
C
C       WRITE THE REACTION NUMBER AND REACTION COEFFIENTS, FOR BOTH
C       REACTANT AND PRODUCT SIDES FOR EACH REACTION
C       THE MAXIMUM OF SPECIES THAT CAN BE HANDLED IS 20

        CHARBG        = ' '

        DO 40 IR = 1, NREACH
          DO 20 IS = 1, NSPECH
             INIT = 1 + 6*(IS-1)
             IFIN = INIT + 2
             CHARBG(INIT:IFIN) = 'ALP'
             INIT = IFIN + 1
             IFIN = INIT + 1
             IF (IS .LT. 10) THEN
                WRITE (CHARBG(INIT:INIT), 1300) IS
             ELSE
                WRITE (CHARBG(INIT:IFIN), 1100) IS
             ENDIF
20        CONTINUE
          WRITE(JCHELP,1400) IR, CHARBG(1:120)

          DO 30 IS = 1, NSPECH
             INIT = 1 + 6*(IS-1)
             IFIN = INIT + 2
             CHARBG(INIT:IFIN) = 'BET'
30        CONTINUE
          WRITE(JCHELP,1400) IR, CHARBG(1:120)
40      CONTINUE
C
C       WRITE THE REACTION NUMBER IR, REACTION CONSTANT TYPE IREACT,
C       PRE-EXPONENTIAL FACTOR (NATURAL LOG VALUE), EXPONENT OF
C       TEMPERATURE AND THE ACTIVATION ENERGY TERM (E/R -- PER DEGREE K)
C       FOR EACH REACTION SO THAT REACTION RATES CAN BE DETERMINED.
```

```
C       IREACT IS A BINARY CODED VARIABLE WHICH INDICATES THE RATE
C       CONSTANTS TO BE READ
C       IF IREACT = 1 READ FORWARD     RATE CONSTANTS
C       IF IREACT = 2 READ BACKWARD    RATE CONSTANTS
C       IF IREACT = 4 READ EQUILIBRIUM RATE CONSTANTS
C       FOR ROGER AND CHINITZ MODEL WRITE THE EQUIVALENCE RATIO PHI
C       IN PREFCH(1) AND MEAN TEMPERATURE (TEMPMN) IN PREFCH(2), USE
C       IREACT=5 AND NEGATIVE VALUE FOR TEMPMN IF A LINEAR SPECIFIC
C       HEAT MODEL (Cp=a+bT) IS DESIRED.
C       FOR LIGHT HILL MODEL WRITE THE CONSTANT PHI IN PREFCH(1),
C       ETA IN EXPFCH(1), THETAD IN ENEFCH(1) AND RHOD IN PREBCH(1).
C
        CHARBG        = ' '
        CHARBG(58:99) = '! REACTION CONSTANTS'

        DO 50 IR = 1, NREACH

           IF (KROGER .EQ. 2) THEN
             IREACT = 5
           ELSE
              WRITE (JTERMO,1500)
              READ  (JTERMI,*) IREACT
           ENDIF

           WRITE(JCHELP,1600) IR, IREACT

           IRF = IAND (IREACT,KL0001)
           IRB = IAND (IREACT,KL0002)
           IRE = IAND (IREACT,KL0004)

           IF (IRF .NE. 0) THEN
             CHARBG(1:36) = 'PREFCH( )   EXPFCH( )   ENEFCH( )'
             IF (KROGER .EQ. 1 .AND. IR .EQ. 1) CHARBG(1:6) = 'PHI EQ'
             IF (KROGER .EQ. 1 .AND. IR .EQ. 2) CHARBG(1:6) = 'TEMPMN'
             WRITE (CHARBG( 8: 9), 1100) IR
             WRITE (CHARBG(21:22), 1100) IR
             WRITE (CHARBG(34:35), 1100) IR
             IF (KROGER .EQ. 1) CHARBG(8:9) = '  '
             IF (KROGER .EQ. 2) THEN
                CHARBG(1:36)='PHI LH       ETA          THETAD'
             ENDIF
             WRITE(JCHELP,1700) CHARBG(1:99)
           ENDIF

           IF (IRB .NE. 0) THEN
             CHARBG(1:36) = 'PREBCH( )   EXPBCH( )   ENEBCH( )'
             WRITE (CHARBG( 8: 9), 1100) IR
             WRITE (CHARBG(21:22), 1100) IR
             WRITE (CHARBG(34:35), 1100) IR
             IF (KROGER .EQ. 2) CHARBG(1:10) = 'RHOD '
             WRITE(JCHELP,1700) CHARBG(1:99)
           ENDIF

           IF (IRE .NE. 0) THEN
             CHARBG(1:36) = 'PREECH( )   EXPECH( )   ENEECH( )'
             WRITE (CHARBG( 8: 9), 1100) IR
             WRITE (CHARBG(21:22), 1100) IR
```

```
            WRITE (CHARBG(34:35), 1100) IR
           _WRITE(JCHELP,1700) CHARBG(1:99)
          ENDIF
50      CONTINUE

        IF (KROGER .EQ. 1) GO TO 70

C       WRITE THE SPECIES NUMBER IS, ATOMIC WEIGHT, SPECIFIC HEATS
C       (CP AND CV IN KJ/KG/K), HEAT OF FORMATION (KJ/KMOL), AND
C       ENTROPY (KJ/KMOL/K) AT  THE REFERENCE CONDITIONS FOR EACH SPECIES

        CHARBG        = ' '
        CHARBG(71:99) = '! SPECIES VALUES '

        DO 60 IS = 1, NSPECH
           CHARBG( 1:39) = 'AMWTCH( )   SPCPCH( )   SPCVCH( )   '
           CHARBG(40:78) = 'HTFMCH( )   ENTRCH( )   SPBSCH( )   '
           WRITE (CHARBG( 8: 9), 1100) IS
           WRITE (CHARBG(21:22), 1100) IS
           WRITE (CHARBG(34:35), 1100) IS
           WRITE (CHARBG(47:48), 1100) IS
           WRITE (CHARBG(60:61), 1100) IS
           WRITE (CHARBG(73:74), 1100) IS
           WRITE(JCHELP,1200) IS,CHARBG(1:99)
60      CONTINUE
C
70      CHARBG        = ' '

C       SEE IF THERE ARE ANY NON-ELEMENTARY REACTIONS, IF SO
C       WRITE THE REACTION NUMBER AND REACTION ORDER COEFFIENTS,
C       FOR BOTH REACTANT AND PRODUCT SIDES FOR EACH REACTION

        IF (KORDER .GT. 0) THEN
          WRITE(JCHELP,1800)
          DO 100 IR = 1, NREACT
            DO 80 IS = 1, NSPECH
              INIT = 1 + 6*(IS-1)
              IFIN = INIT + 2
              CHARBG(INIT:IFIN) = 'ALO'
              INIT = IFIN + 1
              IFIN = INIT + 1
              IF (IS .LT. 10) THEN
                 WRITE (CHARBG(INIT:INIT), 1300) IS
              ELSE
                 WRITE (CHARBG(INIT:IFIN), 1100) IS
              ENDIF
80          CONTINUE
            WRITE(JCHELP,1400) IR, CHARBG(1:120)
            DO 90 IS = 1, NSPECH
              INIT = 1 + 6*(IS-1)
              IFIN = INIT + 2
              CHARBG(INIT:IFIN) = 'BTO'
90          CONTINUE
            WRITE(JCHELP,1400) IR, CHARBG(1:120)
100       CONTINUE
        ENDIF
```

```
      CLOSE (JCHELP)
C     ----------------
C     FORMAT STATEMENTS
C     ----------------
1000  FORMAT(5X,A)
1100  FORMAT(I2)
1200  FORMAT(1X,'IS=',I2,3X,A)
1300  FORMAT(I1)
1400  FORMAT(1X,'IR=',I2,3X,A)
1500  FORMAT(//5X,'INPUT THE REACTION CONSTANT TYPE IREACT (BINARY)'/
     1         10X,'1 : FOR FORWARD  RATE CONSTANTS'/
     2         10X,'2 : FOR BACKWARD RATE CONSTANTS'/
     3         10X,'4 : EQUILIRIUM   RATE CONSTANTS'/
     4         10X,'8 : DECIDE ABOUT EACH REACTION SEPERATELY'/  )
1600  FORMAT(1X,'IR = ',I2,' IREACT = ',I2)
1700  FORMAT(10X,A)
1800  FORMAT(5X,'NREACT',61X,'!  # NON-ELEMENTARY REACTIONS')

C     RETURN
      END
```

C2INIT

```
      SUBROUTINE C2INIT
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'HEXCOD.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
      INCLUDE 'PRCOMN.INC'
C
      DIMENSION      AKEQ(MREACH)   ,   TEMPER(3 )     ,
     1               RHS (3     )   ,   AMAT  (3,3)     ,
     2               SOLN(3     )
C
C*****************************************************************
C
C     THIS SUBROUTINE READS THE CONSTANTS FOR CHEMISTRY FROM UNIT
C     JREADC.
C
C*****************************************************************
C
C     GET THE VALUES SET BY GETKY2 SUBROUTINE OR THE DEFAULT VALUES
C
      TREFCH    = APASKY( 8)
      PRESCH    = APASKY(10)
      TEMPER(1) = APASKY(12)
      TEMPER(2) = APASKY(13)
      TEMPER(3) = APASKY(14)
```

```
      TRIGCH    = APASKY(25)

      NREACH    = IPASKY( 1)
      NSPECH    = IPASKY( 2)
      KORDER    = IPASKY( 4)
      NINRCH    = IPASKY(18)
      NEQBAS    = 4

C     CHECK FOR ERRORS IN PARAMETER STATEMENTS

      IF (NREACH .GT. MREACH) THEN
        ZER1 = NREACH
        ZER2 = MREACH
        CALL ERRORM (2,'C2INIT','NREACH',ZER1,'MREACH',ZER2,JPRINT,
     1             'PARAMETER ERROR IN REACTION NUMBERS')
      ENDIF

      IF (NSPECH .GT. MSPECH) THEN
        ZER1 = NSPECH
        ZER2 = MSPECH
      CALL ERRORM (2,'C2INIT','NSPECH',ZER1,'MSPECH',ZER2,JPRINT,
     1             'PARAMETER ERROR IN SPECIES NUMBERS')
      ENDIF

C
C     ---------------------------
C     EXPLANATION OF NOMENCLATURE
C     ---------------------------
C
C     NREACH      = NUMBER OF REACTIONS
C     NSPECH      = NUMBER OF SPECIES
C     NINRCH      = NUMBER OF INERT SPECIES
C     IALPCH(S,R) = REACTANT COEFFIENT FOR SPECIES S IN REACTION R
C     IBETCH(S,R) = PRODUCT  COEFFIENT FOR SPECIES S IN REACTION R
C     IALOCH(S,R) = ORDER OF REACTION  FOR SPECIES S IN REACTION R
C     IBTOCH(S,R) = ORDER OF REACTION  FOR SPECIES S IN REACTION R
C     ENTRCH(S)   = REFERENCE ENTROPY  FOR SPECIES S, KJ/KMOL/K
C     AMWTCH(S)   = ATOMIC WEIGHT      FOR SPECIES S
C     FMHTCH(S)   = HEAT OF FORMATION  FOR SPECIES S IN KJ/KMOL
C                   AT THE REFERENCE TEMPERATURE AND PRESSURE
C     SPCPCH(S)   = CONSTANT PRESSURE SPECIFIC HEAT FOR S, KJ/KMOL/K
C     SPCVCH(S)   = CONSTANT  VOLUME  SPECIFIC HEAT FOR S, KJ/KMOL/K
C     PREFCH(S)   = PRE-EXPONENTIAL FACTOR  FOR REACTION R ( FORWARD)
C     PREBCH(S)   = PRE-EXPONENTIAL FACTOR  FOR REACTION R (BACKWARD)
C     EXPFCH(S)   = EXPONENT OF TEMPERATURE FOR REACTION R ( FORWARD)
C     EXPBCH(S)   = EXPONENT OF TEMPERATURE FOR REACTION R (BACKWARD)
C     ENEFCH(S)   = ENERGY TERM (E/R)       FOR REACTION R ( FORWARD)
C     ENEBCH(S)   = ENERGY TERM (E/R)       FOR REACTION R (BACKWARD)
C     IDBGCH      = DEBUG PARAMETER FOR CHEMISTRY
C                   -1 : WRITE EACH STEP
C     JDEBUG      = DEBUG UNIT FOR CHEMISTRY
C     TREFCH      = REFERENCE TEMPERATURE FOR THE CHEMICAL TERMS (298 K)
C     PRESCH      = REFERENCE PRESSURE FOR THE CHEMICAL TERMS (0.1 MPA)

C     INITIALIZE THE REACTION COEFFICIENTS

      DO 10 IS = 1, MSPECH
```

```
      DO 10 IR = 1, MREACH
          IALPCH(IS,IR) = 0
          IBETCH(IS,IR) = 0
10    CONTINUE

C
C     READ INPUTS FROM INPUT CHEMISTRY FILE; READ PARAMETER FOR
C     DEBUG PRINTING FIRST
C
      READ(JREADC,*) IDBGCH
C
C     READ THE REFERENCE AND FREE STREAM MASS FRACTIONS
C     (YSPECH(S) AND DPENFR(S)) FOR ALL THE SPECIES S

      DO 20 ISP = 1, NSPECH
         READ(JREADC,*) IS, YSPECH(IS), DPENFR(IS)
         YSPEPR(IS) = YSPECH(IS)
         IF (IDBGCH .EQ. -1)
     1        WRITE (JDEBUG,*) IS, YSPECH(IS), DPENFR(IS)
20    CONTINUE
C
C     KROGER DENOTES THE SPECIAL TYPE OF REACTION MODEL TO BE USED.
C         KROGER = 1   : USE ROGER AND CHINITZ MODEL
C         KROGER = 2   : USE LIGHT HILL SINGLE DISSOCIATING GAS MODEL
C         KROGER = 3   : USE A SINGLE NON-REACTING GAS
C
C     SET UP THE CONSTANTS FOR ROGER AND CHINITZ MODEL, FOR THIS
C     MODEL THE SPECIES MUST BE ORDERED AS O2, H2O, H2, OH AND N2
C
      IF (KROGER .EQ. 1) THEN
        NSPECH = 5
        NREACH = 2
        NINRCH = 1
      ENDIF
C
C     FOR THE LIGHT HILL GAS MODEL THE SPECIES ARE ORDERED A, A2
C
      IF (KROGER .EQ. 2) THEN
        NSPECH = 2
        NREACH = 1
        NINRCH = 0
      ENDIF
C
      IF (KROGER .EQ. 3) THEN
        NSPECH = 1
        NREACH = 0
        NINRCH = 0
        TREFCH = 0.
      ENDIF
C
      IF (IDBGCH .EQ. -1) THEN
          WRITE(JDEBUG,1000)
          WRITE(JDEBUG,1100)
          WRITE(JDEBUG,1200)
          WRITE(JDEBUG,1300) NREACH, NSPECH
          WRITE(JDEBUG,1400)
      ENDIF
```

```
C       READ THE REACTION NUMBER AND REACTION COEFFICIENTS, FOR BOTH
C       REACTANT AND PRODUCT SIDES FOR EACH REACTION

        DO 40 IR = 1, NREACH
            READ(JREADC,*) IRP, (IALPCH(IS,IR),IS=1,NSPECH)
            READ(JREADC,*) IRP, (IBETCH(IS,IR),IS=1,NSPECH)
            DO 30 IS = 1, NSPECH
              BMIACH(IS,IR) = IBETCH(IS,IR) - IALPCH(IS,IR)
30          CONTINUE
            IF (IDBGCH .EQ. -1) THEN
               WRITE(JDEBUG,1500) IR, (IALPCH(IS,IR),IS=1,NSPECH)
               WRITE(JDEBUG,1500) IR, (IBETCH(IS,IR),IS=1,NSPECH)
               WRITE(JDEBUG,1600)
            ENDIF

40      CONTINUE

C       READ THE REACTION NUMBER IR, REACTION CONSTANT TYPE IREACT,
C       PRE-EXPONENTIAL FACTOR (NATURAL LOG VALUE), EXPONENT OF
C       TEMPERATURE AND THE ACTIVATION ENERGY TERM (E/R -- PER DEGREE K)
C       FOR EACH REACTION SO THAT REACTION RATES CAN BE DETERMINED.
C       IREACT IS A BINARY CODED VARIABLE WHICH INDICATES THE RATE
C       CONSTANTS TO BE READ
C       IF IREACT = 1 READ FORWARD     RATE CONSTANTS
C       IF IREACT = 2 READ BACKWARD    RATE CONSTANTS
C       IF IREACT = 4 READ EQUILIBRIUM RATE CONSTANTS
C       FOR ROGER AND CHINITZ MODEL READ THE EQUIVALENCE RATIO PHI
C       IN PREFCH(1) AND MEAN TEMPERATURE (TEMPMN) IN PREFCH(2), USE
C       IREACT=5 AND NEGATIVE VALUE FOR TEMPMN IF A LINEAR SPECIFIC
C       HEAT MODEL (Cp=a+bT) IS DESIRED.
C       FOR LIGHT HILL MODEL READ THE CONSTANT PHI IN PREFCH(1),
C       ETA IN EXPFCH(1), THETAD IN ENEFCH(1) AND RHOD IN PREBCH(1).
C
        IF (IDBGCH .EQ. -1) WRITE(JDEBUG,2400)

        DO 50 IR = 1, NREACH

            READ(JREADC,*) IRP, IREACT

            IRF = IAND (IREACT,KLOOO1)
            IRB = IAND (IREACT,KLOOO2)
            IRE = IAND (IREACT,KLOOO4)

C           FORWARD RATE CONSTANTS ?

            IF (IRF .NE. O) THEN
               READ(JREADC,*) TERM1, TERM2, TERM3
               IF (IDBGCH .EQ. -1) THEN
                 WRITE(JDEBUG,2600) IRP, IREACT, TERM1, TERM2, TERM3
               ENDIF
               PREFCH(IR) = TERM1
               EXPFCH(IR) = TERM2
               ENEFCH(IR) = TERM3
               IF (IRB .EQ. O .AND. IRE .EQ. O) PREBCH(IR) = -99.
            ENDIF
```

```
C          BACKWARD RATE CONSTANTS ?

           IF (IRB .NE. 0) THEN
              READ(JREADC,*) TERM1, TERM2, TERM3
              IF (IDBGCH .EQ. -1) THEN
                WRITE(JDEBUG,2600) IRP, IREACT, TERM1, TERM2, TERM3
              ENDIF
              PREBCH(IR) = TERM1
              EXPBCH(IR) = TERM2
              ENEBCH(IR) = TERM3
              IF (IRF .EQ. 0 .AND. IRE .EQ. 0) PREFCH(IR) = -99.
              IF (IRF .NE. 0 .AND. IRE .EQ. 0) THEN
                 PREECH(IR) = PREFCH(IR) - PREBCH(IR)
                 EXPECH(IR) = EXPFCH(IR) - EXPBCH(IR)
                 ENEECH(IR) = ENEFCH(IR) - ENEBCH(IR)
              ENDIF
           ENDIF

C          EQUILIBRIUM RATE CONSTANTS ?

           IF (IRE .NE. 0) THEN
              READ(JREADC,*) TERM1, TERM2, TERM3
              IF (IDBGCH .EQ. -1) THEN
                WRITE(JDEBUG,2600) IRP, IREACT, TERM1, TERM2, TERM3
              ENDIF
              PREECH(IR) = TERM1
              EXPECH(IR) = TERM2
              ENEECH(IR) = TERM3
              IF (IRB .EQ. 0) THEN
                 PREBCH(IR) = PREFCH(IR) - PREECH(IR)
                 EXPBCH(IR) = EXPFCH(IR) - EXPECH(IR)
                 ENEBCH(IR) = ENEFCH(IR) - ENEECH(IR)
              ENDIF
              IF (IRF .EQ. 0) THEN
                 PREFCH(IR) = PREBCH(IR) + PREECH(IR)
                 EXPFCH(IR) = EXPBCH(IR) + EXPECH(IR)
                 ENEFCH(IR) = ENEBCH(IR) + ENEECH(IR)
              ENDIF
           ENDIF
50      CONTINUE

        IF (KROGER .EQ. 1) THEN
C
C          PHICR < 0.1 IS NOT TOLERABLE IN C2ROCH, BUT IS OK FOR C2RINT
           PHICR = PREFCH(1)
           CALL C2ROCH
           CALL C2RINT (PHICR,YO2,YH2,YN2)
           PREBCH(1) = PREFCH(1) - PREECH(1)
           PREBCH(2) = PREFCH(2) - PREECH(2)
           YSPEPR(1) = YO2
           YSPECH(1) = YO2
           YSPEPR(2) = 0.
           YSPECH(2) = 0.
           YSPEPR(3) = YH2
           YSPECH(3) = YH2
           YSPEPR(4) = 0.
           YSPECH(4) = 0.
```

```
            YSPEPR(5) = YN2
            YSPECH(5) = YN2
            IF (YH2 .NE. 0.) THEN
               WRITE(6,*) ' C2INIT PHICR YO2 YH2 YN2',
     1                           PHICR,YO2,YH2,YN2
            ENDIF
C           SKIP THE SPECIFIC HEAT AND SOME OTHER DATA
            GO TO 70
          ENDIF
C
          IF (IDBGCH .EQ. -1) THEN
            WRITE(JDEBUG,2000)
            WRITE(JDEBUG,2100)
          ENDIF

C         READ THE SPECIES NUMBER IS, ATOMIC WEIGHT, SPECIFIC HEATS
C         (CP AND CV IN KJ/KG/K), HEAT OF FORMATION (KJ/KMOL), AND
C         ENTROPY (KJ/KMOL/K) AT  THE REFERENCE CONDITIONS FOR EACH SPECIES

          DO 60 IS = 1, NSPECH
              READ(JREADC,*) ISP, AMWTCH(IS), SPCPCH(IS), SPCVCH(IS),
     1                           FMHTCH(IS), ENTRCH(IS), SPBSCH(IS)
              SPCPCH(IS) = 1000.*SPCPCH(IS)
              SPBSCH(IS) = 1000.*SPBSCH(IS)
              SPCVCH(IS) = 1000.*SPCVCH(IS)
              FMHTCH(IS) = 1000.*FMHTCH(IS)/AMWTCH(IS)
              ENTRCH(IS) = 1000.*ENTRCH(IS)

              IF (IDBGCH .EQ. -1) THEN
                WRITE(JDEBUG,2200) IS, AMWTCH(IS), SPCPCH(IS),
     1                     SPCVCH(IS), FMHTCH(IS), ENTRCH(IS), SPBSCH(IS)
              ENDIF

60        CONTINUE
C
70        IF (KORDER .GT. 0) THEN
C           READ THE NUMBER OF NON-ELEMENTARY REACTIONS
            READ(JREADC,*) NREACT
            DO 80 IR = 1, NREACT
C             READ THE REACTION NUMBER AND REACTION ORDER COEFFICIENTS,
C             FOR BOTH REACTANT AND PRODUCT SIDES FOR EACH REACTION
              READ(JREADC,*) IRP, (IALOCH(IS,IR),IS=1,NSPECH)
              READ(JREADC,*) IRP, (IBTOCH(IS,IR),IS=1,NSPECH)
80          CONTINUE
          ELSE
            DO 90 IR = 1, NREACH
              DO 90 IS = 1, NSPECH
                IALOCH(IS,IR) = IALPCH(IS,IR)
                IBTOCH(IS,IR) = IBETCH(IS,IR)
90          CONTINUE
          ENDIF

          DO 120 IR = 1, NREACH
            IF (PREBCH(IR) .EQ. -99.) THEN
               DO 100 IT = 1, 3
                 TEMP = TEMPER(IT)
                 AT   = LOG(TEMP)
```

```
              RT   = 1./TEMP
              ALKF = PREFCH(IR) + EXPFCH(IR)*AT - ENEFCH(IR)*RT
C             GET THE LOG OF EQUILIBRIUM CONSTANT FOR REACTION IR
C             CALL C2KCRE(TEMP, AKEQ, IR)
              RHS(IT) = ALKF - AKEQ(IR)
              AMAT(IT,1) = 1.
              AMAT(IT,2) = AT
              AMAT(IT,3) = -RT
100           CONTINUE
              CALL GAUSS2(AMAT,RHS,SOLN,3,3)
              PREBCH(IR) = SOLN(1)
              EXPBCH(IR) = SOLN(2)
              ENEBCH(IR) = SOLN(3)
            ENDIF

            IF (PREFCH(IR) .EQ. -99.) THEN
              DO 110 IT = 1, 3
                TEMP = TEMPER(IT)
                AT   = LOG(TEMP)
                RT   = 1./TEMP
                ALKB = PREBCH(IR) + EXPBCH(IR)*AT - ENEBCH(IR)*RT
C               CALL C2KCRE(TEMP, AKEQ, IR)
                RHS(IT) = ALKB + AKEQ(IR)
                AMAT(IT,1) = 1.
                AMAT(IT,2) = AT
                AMAT(IT,3) = -RT
110           CONTINUE
              CALL GAUSS2(AMAT,RHS,SOLN,3,3)
              PREFCH(IR) = SOLN(1)
              EXPFCH(IR) = SOLN(2)
              ENEFCH(IR) = SOLN(3)
            ENDIF
120       CONTINUE

          CLOSE (JREADC)
C
C         PRINT OUT PARAMETERS
C
          IF (IDBGCH .NE. 1 .AND. IDBGCH .LT. 1000)  RETURN

          WRITE(JDEBUG,1000)
          WRITE(JDEBUG,1100)
          WRITE(JDEBUG,1200)

          WRITE(JDEBUG,1300) NREACH, NSPECH
          WRITE(JDEBUG,1400)

          DO 130 IR = 1, NREACH
              WRITE(JDEBUG,1500) IR, (IALPCH(IS,IR),IS=1,NSPECH)
              WRITE(JDEBUG,1500) IR, (IBETCH(IS,IR),IS=1,NSPECH)
              WRITE(JDEBUG,1600)
130       CONTINUE

          IF (KORDER .EQ. 0) THEN
            WRITE(JDEBUG,1700)
          ELSE
            WRITE(JDEBUG,1800)
```

```
          WRITE(JDEBUG,1900)
          DO-140 IR = 1, NREACH
             WRITE(JDEBUG,1500) IR, (IALOCH(IS,IR),IS=1,NSPECH)
             WRITE(JDEBUG,1500) IR, (IBTOCH(IS,IR),IS=1,NSPECH)
140       CONTINUE
        ENDIF

        WRITE(JDEBUG,2000)
        WRITE(JDEBUG,2100)

        DO 150 IS = 1, NSPECH
           WRITE(JDEBUG,2200) IS, AMWTCH(IS), SPCPCH(IS),
     1                SPCVCH(IS), FMHTCH(IS), ENTRCH(IS)
150     CONTINUE

        WRITE(JDEBUG,2300)

        DO 160 IR = 1, NREACH
           WRITE(JDEBUG,2500) IR, PREFCH(IR), EXPFCH(IR), ENEFCH(IR)
           WRITE(JDEBUG,2500) IR, PREBCH(IR), EXPBCH(IR), ENEBCH(IR)
           WRITE(JDEBUG,2500) IR, PREECH(IR), EXPECH(IR), ENEECH(IR)
           WRITE(JDEBUG,1600)
160     CONTINUE

        WRITE(JDEBUG,2700) TREFCH, PRESCH

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM C2INIT' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'NUMBER OF REACTIONS = ',I5,5X,
     1        5X,'NUMBER OF SPECIES   = ',I5//)
1400    FORMAT(5X,'REACTION    REACTION COEFFICIENTS'/)
1500    FORMAT(5X,I5,5X,20I5)
1600    FORMAT(/)
1700    FORMAT(/5X,'-----ALL THE REACTIONS ARE ELEMENTARY -----'/)
1800    FORMAT(/5X,'-----SOME REACTIONS ARE NON-ELEMENTARY -----'/)
1900    FORMAT(5X,'REACTION       REACTION ORDER COEFFICIENTS'/)
2000    FORMAT(/5X,'-----PROPERTIES OF SPECIES-----'/)
2100    FORMAT(8X,'SPECIES',5X,'MOL WT',11X,'CP -- J/KG/K',5X,
     1        'CV -- J/KG/K',5X,'HT FM J/KG  ',5X,
     2        'ENTROPY J/KMOL/K'/)
2200    FORMAT(5X,I5,5X,6E17.6)
2300    FORMAT(/5X,'-----PROPERTIES OF REACTIONS-----'//
     1         5X,'REACTION',2X,'PRE-EXPO FAC',5X,'EXPONENT',
     2         9X,'ENERGY TERM')
2400    FORMAT(/5X,'-----PROPERTIES OF REACTIONS-----'//
     1         5X,'REACTION',2X,'IREACT',4X,'TERM1',12X,'TERM2',
     2        12X,'TERM3')
2500    FORMAT(5X,I5,3E17.6)
2600    FORMAT(5X,I5,5X,I5,3E17.6)
2700    FORMAT(//5X,'REFERENCE TEMPERATURE =',E15.6,5X,
     1          5X,'REFERENCE PRESSURE    =',E15.6//)
```

```
      RETURN
      END
```

C2KCRE

```
      SUBROUTINE C2KCRE (TEMP, AKEQ, ITYPE)
C
      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'
C
      DIMENSION AKEQ(MREACH), GIBBS(MSPECH)
      DATA KOUNT /0/
C
C*********************************************************************
C
C     THIS SUBROUTINE COMPUTES THE REACTION CONSTANTS KP AND KC FOR A
C     NUMBER OF REACTIONS AT A GIVEN TEMPERATURE TEMP IN DEGREE K.
C
C*********************************************************************
C
C     GIBBS(S) IS THE SPECIFIC MOLAL GIBBS FUNCTION FOR SPECIES S
C     DELGIB IS THE TOTAL GIBBS FUNCTION CHANGE FOR A REACTION R
C     SUMCOF IS THE SUM OF THE COEFFIENTS (DELTA_N) USEFUL IN THE
C     COMPUTATION OF EQUILIBRIUM CONSTANT
C     AKPR IS THE EQUILIBRIUM CONSTANT BASED ON PARTIAL PRESSURES
C     AKEQ(R) IS THE EQUILIBRIUM CONSTANT BASED ON CONCENTRATIONS
C     ITYPE IS THE PARAMETER INDICATING WHETHER EQUILIBRIUM CONSTANTS
C     HAVE TO BE DETERMINED FOR ALL REACTIONS (SET ITYPE < 0) OR FOR
C     SPECIFIC REACTIONS (SET ITYPE = IR -- THE REACTION NUMBER)
C
      UGASFL = 8.31434E03
      KOUNT  = KOUNT + 1
      DO 10 IS = 1, NSPECH
        GIBBS(IS) = FMHTCH(IS)*AMWTCH(IS) - TEMP*ENTRCH(IS)
     1              + SPCPCH(IS)*AMWTCH(IS)*(TEMP-TREFCH)
     2              + 0.5*SPBSCH(IS)*AMWTCH(IS)*(TEMP**2-TREFCH**2)
     3              - TEMP*SPCPCH(IS)*AMWTCH(IS)*LOG(TEMP/TREFCH)
     4              - TEMP*SPBSCH(IS)*AMWTCH(IS)*(TEMP-TREFCH)
C
10    CONTINUE
C
C     COMPUTE THE (LOG OF) EQUILIBRIUM CONSTANTS FOR ALL THE REACTIONS
C
      IF (ITYPE .LT. 0) THEN
        DO 30 IR = 1, NREACH
          DELGIB = 0.
          SUMCOF = 0.
          DO 20 IS = 1, NSPECH
            DELGIB = DELGIB + BMIACH(IS,IR)*GIBBS(IS)
            SUMCOF = SUMCOF - BMIACH(IS,IR)
```

```
20          CONTINUE
            AKPR     = -DELGIB/TEMP/UGASFL
            AKEQ(IR) = AKPR + SUMCOF*LOG(UGASFL*TEMP/PRESCH)
30        CONTINUE
          GOTO 50
        ENDIF
C
C       COMPUTE THE EQUILIBRIUM CONSTANTS FOR ALL REACTION IR
C
        IR     = ITYPE
        DELGIB = 0.
        SUMCOF = 0.
          DO 40 IS = 1, NSPECH
            DELGIB = DELGIB + BMIACH(IS,IR)*GIBBS(IS)
            SUMCOF = SUMCOF - BMIACH(IS,IR)
40        CONTINUE
        AKPR     = -DELGIB/TEMP/UGASFL
        AKEQ(IR) = AKPR + SUMCOF*LOG(UGASFL*TEMP/PRESCH)
50      CONTINUE
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGCH .NE. 2 .AND. IDBGCH .LT. 1000)  RETURN
        IF (KOUNT  .NE. 1 ) GOTO 60
C
        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
60      WRITE(JDEBUG,1300)
C
        DO 70 IS = 1, NSPECH
           WRITE(JDEBUG,1400) IS,GIBBS(IS)
70      CONTINUE
C
        WRITE(JDEBUG,1500)
C
        IR = ITYPE
C
        IF (ITYPE .LT. 0) THEN
           DO 80 IR = 1, NREACH
              WRITE(JDEBUG,1600) IR,TEMP,AKEQ(IR)
80         CONTINUE
        ELSE
           WRITE(JDEBUG,1600) IR,TEMP,AKEQ(IR)
        ENDIF
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM C2KCRE' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT (' GIBBS FUNCTION FOR VARIOUS SPECIES :'/
     1         5X,'SPECIES',3X,'GIBBS FUNCTION')
1400    FORMAT(5X,I5,5X,E15.7)
1500    FORMAT(/5X,'REACTION',10X,'TEMPERATURE',4X,'LOG OF EQUIL CONS'/)
```

```
1600    FORMAT( 5X,I5,10X,2E15.6)
C           -
        RETURN
        END
```

C2PLOT

```
        PROGRAM C2PLOT

        PARAMETER (MNODG2 = 1000, MLINE=10)
        DIMENSION TEMP$(MNODG2), CP$(MNODG2)

        DIMENSION N$(MLINE), IOPT$(MLINE)
        CHARACTER FILNAM*40, MTITLE*80, PLTITL*96

C***********************************************************************
C
C       THIS PROGRAM PLOTS THE SPECIFIC HEAT AT CONSTANT PRESSURE, BOTH
C       THE ORIGINAL DATA AND THE LINEAR REGRESSION FIT.  THIS PROGRAM
C       FIRST READS THE TWO KINDS OF DATA FROM THE S.DATn FILES WHICH
C       MAY HAVE BEEN CREATED BY C2SPCA.FOR FOR A GIVEN NUMBER OF SPECIES
C
C***********************************************************************

        JTERMO = 6
        JTERMI = 5
        OFFSET = 10.
        JOUNT  = 5

        MTITLE = ' '
        CALL GR_INIT(JTERMI,JTERMO,MTITLE)
        PLTITL   = ' '
        KOUNTM = 0
        OFSETM = 0.
        NLINE    = 0
        WRITE(JTERMO,*) ' INPUT TOTAL NUMBER OF SPECIES'
        READ (JTERMI,*) NSPECH

        DO 100 ILINE = 1, NSPECH
           FILNAM=' '
           FILNAM(1:5)='S.DAT'
           WRITE(FILNAM(6:6),1778) ILINE
1778       FORMAT(I1)
           OPEN (UNIT=48,FILE=FILNAM,STATUS='OLD',READONLY)
           READ(48,*) KOUNT
           NLINE    = NLINE + 1
           IOPT$(NLINE) = 12
           N$(NLINE)    = KOUNT
           DO 1134 I = 1, KOUNT
             KOUNTM = KOUNTM + 1
             READ(48,1005) TEMP,CP,CPL
             TEMP$(KOUNTM) = TEMP
             CP$(KOUNTM) = CP + OFSETM
```

```
1134        CONTINUE
            OFSETM = OFSETM + OFFSET
1005        FORMAT(3E15.7)
            CLOSE(48)
100      CONTINUE

         OFSETM = 0.

         DO 200 ILINE = 1, NSPECH
            FILNAM=' '
            FILNAM(1:5)='S.DAT'
            WRITE(FILNAM(6:6),1778) ILINE
            OPEN (UNIT=48,FILE=FILNAM,STATUS='OLD',READONLY)
            READ(48,*) KOUNT
            NLINE    = NLINE + 1
            IOPT$(NLINE) = 2
            N$(NLINE)    = KOUNT
            DO 1135 I = 1, KOUNT
              KOUNTM = KOUNTM + 1
              READ(48,1005) TEMP,CP,CPL
              TEMP$(KOUNTM) = TEMP
              CP$(KOUNTM) = CPL + OFSETM
1135        CONTINUE
            OFSETM = OFSETM + OFFSET
            CLOSE(48)
200      CONTINUE

         INDGR    = 21

         CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,TEMP$, CP$,N$)
         STOP
         END
```

C2PONT

```
         SUBROUTINE C2PONT
C
         INCLUDE 'PRECIS.INC'
         INCLUDE 'PARMV2.INC'
         INCLUDE 'CHCOMN.INC'
         INCLUDE 'IOCOMN.INC'

         LOGICAL  INERT

C*****************************************************************
C
C        THIS SUBROUTINE SETS THE CHEMISTRY POINTER SYSTEM FOR ALL THE
C        REACTIONS.  THAT IS, IT SETS THE NUMBER OF SPECIES IN EACH
C        REACTION NSRKCH(IR) AND A TABLE ITABCH(IS,IR) OF SPECIES
C        NUMBERS INVOLVED IN THE KINETIC (OR EQUILILIBRIUM) REACTIONS
C        FOR THE SPECIES IS IN THE REACTION IR.
C
C*****************************************************************
```

```
C       NONINR COUNTS THE NON-INERT SPECIES
C
        DO 20 IR = 1, NREACH

          NONINR = 0

          DO 10 IS = 1, NSPECH

            INERT = IALPCH(IS,IR) .EQ. 0 .AND. IBETCH(IS,IR) .EQ. 0

            IF (.NOT. INERT ) THEN
               NONINR = NONINR + 1
               ITABCH(NONINR,IR) = IS
            ENDIF

10        CONTINUE

          NSRKCH(IR) = NONINR

20      CONTINUE
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGCH .NE. 5 .AND. IDBGCH .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)

        DO 40 IR = 1, NREACH
           WRITE(JDEBUG,1300) IR, NSRKCH(IR)
           DO 30 IS = 1, NSRKCH(IR)
              WRITE(JDEBUG,1400) ITABCH(IS,IR)
30         CONTINUE
40      CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM C2PONT' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(/5X,'REACTION #',I2,5X,'SPECIES IN THIS REACTION=',I2/
     1          5X,'SPECIES COEFFICIENTS ARE :')
1400    FORMAT(5X, 60I3)

        RETURN
        END
```

C2RINT

```
      SUBROUTINE C2RINT (PHI, YO2, YH2, YN2)

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'IOCOMN.INC'

C*********************************************************************
C
C     THIS SUBROUTINE COMPUTES THE INITIAL CONCENTRATIONS FOR A PRE-
C     MIXED FLOW FOR ROGER AND CHINITZ MODEL, I.E., FOR A SCRAMJET
C     CALCULATION OF HYDROGEN FUEL IN AIR FOR A GIVEN VALUE OF
C     EQUIVALENCE RATIO PHI.  THE STOICHIOMETRIC REACTION OF HYDROGEN
C     IN AIR IS
C
C          2 H   +  ( O   + 3.76 N  )  <===>  2 H O  +  3.76 N
C            2       2         2              2           2
C
C
C*********************************************************************

C     ASSIGN THE MOLECULAR WEIGHTS
C
      AMWTO2 = AMWTCH(1)
      AMWTH2 = AMWTCH(3)
      AMWTN2 = AMWTCH(5)

C     MOLAR RATIO OF NITROGEN AND OXYGEN IN AIR

      RN2BO2 = 3.76

      AMTO2  = AMWTO2
      AMTN2  = AMWTN2*RN2BO2
      AMTAIR = AMTN2 + AMTO2
      AMTH2S = AMWTH2*2.

      FBAIRS = AMTH2S/AMTAIR
      FBAIR  = PHI*FBAIRS
      AMTH2  = FBAIR*AMTAIR
      TOTALM = AMTH2 + AMTAIR

      YO2    = AMTO2/TOTALM
      YH2    = AMTH2/TOTALM
      YN2    = AMTN2/TOTALM
      YH2O   = 0.
      YOH    = 0.
      YTOTAL = YO2 + YH2 + YN2
C
C     PRINT OUT PARAMETERS
C
      IF (IDBGCH .NE. 4 .AND. IDBGCH .LT. 1000)  RETURN

      WRITE(JDEBUG,1000)
      WRITE(JDEBUG,1100)
```

```
      WRITE(JDEBUG,1200)
      WRITE(JDEBUG,1300)
      WRITE(JDEBUG,1400) PHI,FBAIRS,FBAIR,YH2,YO2,YN2,YTOTAL

C     -----------------
C     FORMAT STATEMENTS
C     -----------------

1000  FORMAT(//10X,'----------------------' )
1100  FORMAT(  10X,'DEBUG PRINT FROM C2RINT' )
1200  FORMAT(  10X,'----------------------'/)
1300  FORMAT(3X,'PHI',12X,'FUEL/AIR STOI',2X,'FUEL/AIR',7X,'YH2',
     1        12X,'YO2',12X,'YN2',12X,'YTOTAL'/)
1400  FORMAT (7E15.6)

      RETURN
      END
```

C2ROCH

```
      SUBROUTINE C2ROCH

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'IOCOMN.INC'

      CHARACTER*4 SPNAME(5)

C***********************************************************************
C
C     THIS SUBROUTINE COMPUTES THE PRE-EXPONENTIAL FACTORS FOR THE
C     ROGER AND CHINITZ MODEL FOR A GIVEN VALUE OF EQUIVALENCE RATIO
C     PHI.  IT ALSO SETS UP THE PROPERTIES OF THE SPECIES FOR A GIVEN
C     VALUE OF THE MEAN TEMPERATURE TEMPMN
C
C***********************************************************************
C
C     DEFINE THE MOLAL SPECIFIC HEAT OF SPECIES IN KJ/KMOL/K
C     REFERENCE : VAN WYLEN PAGES 683-684 FOR TEMPERATURES BETWEEN
C     300 AND 3500 KELVIN.

      CPN2 (T)=39.060 - 512.79/T**1.5 + 1072.7/T**2 - 820.4/T**3
      CPO2 (T)=37.432 + 0.020102*T**1.5 - 178.57/T**1.5 + 236.88/(T*T)
      CPH2 (T)=56.505 - 702.74/T**0.75 + 1165./T - 560.7/T**1.5
      CPOH (T)=81.546 - 59.350*T**0.25 + 17.329*T**0.75 - 4.266*T
      CPH2O(T)=143.05 - 183.54*T**0.25 + 82.751*T**0.5 - 3.6989*T

C     SET EQUIVALENCE RATIO PHI AND MEAN TEMPERATURE TEMPMN

      PHI    = PREFCH(1)
      TEMPMN = PREFCH(2)
      IF (PHI .LT. 0.1) PHI = 0.1
```

```
      IF (PHI .GT. 2.0) PHI = 2.0
      THE    = ABS(TEMPMN)/100.
      RPHI   = 1./PHI
      TENLOG = LOG(10.)

      A1PHI  = 8.917*PHI + 31.433*RPHI - 28.950
      A2PHI  =-0.833*PHI +  1.333*RPHI +  2.000

      PREFCH(1) = LOG(A1PHI) + 44.*TENLOG
      PREFCH(2) = LOG(A2PHI) + 58.*TENLOG

      EXPFCH(1) = -10.
      EXPFCH(2) = -13.

      ENEFCH(1) = 2448.42
      ENEFCH(2) = 21389.03

C     THE SPECIES ARE ORDERED AS O2, OH, H2, H2O

      SPNAME(1) = 'O2  '
      SPNAME(2) = 'OH  '
      SPNAME(3) = 'H2  '
      SPNAME(4) = 'H2O '
      SPNAME(5) = 'N2  '

C     MOLECULAR WEIGHTS

      AMWTCH(1) = 31.999
      AMWTCH(2) = 17.008
      AMWTCH(3) =  2.016
      AMWTCH(4) = 18.015
      AMWTCH(5) = 28.013

C     HEAT OF FORMATION IN J/KMOL

      FMHTCH(1) = 0.
      FMHTCH(2) = 39463.*1000./AMWTCH(2)
      FMHTCH(3) = 0.
      FMHTCH(4) = -241827.*1000./AMWTCH(4)
      FMHTCH(5) = 0.

C     REFERENCE ENTROPY IN J/KMOL/K

      ENTRCH(1) = 205.142*1000.
      ENTRCH(2) = 183.703*1000.
      ENTRCH(3) = 130.684*1000.
      ENTRCH(4) = 188.833*1000.
      ENTRCH(5) = 191.611*1000.

C     SPECIFIC HEATS AT CONSTANT PRESSURE IN KJ/KMOL/K

      IF (TEMPMN .GT. 0.) THEN
         SPCPCH(1) = CPO2 (THE)
         SPCPCH(2) = CPOH (THE)
         SPCPCH(3) = CPH2 (THE)
         SPCPCH(4) = CPH2O(THE)
         SPCPCH(5) = CPN2 (THE)
```

```
            SPBSCH(1) = 0.
            SPBSCH(2) = 0.
            SPBSCH(3) = 0.
            SPBSCH(4) = 0.
            SPBSCH(5) = 0.
          ELSE
            SPCPCH(1) = 30.559
            SPCPCH(2) = 28.071
            SPCPCH(3) = 27.290
            SPCPCH(4) = 32.469
            SPCPCH(5) = 29.282
            SPBSCH(1) = 0.34485E-2
            SPBSCH(2) = 0.30943E-2
            SPBSCH(3) = 0.33530E-2
            SPBSCH(4) = 0.86358E-2
            SPBSCH(5) = 0.30233E-2
          ENDIF

C         UNIVERSAL GAS CONSTANT IN KJ/KMOL/K

          UGASCO=8.31434

          DO 10 IS = 1, NSPECH
C           SPECIFIC HEATS AT CONSTANT VOLUME IN KJ/KMOL/K
            SPCVCH(IS) = SPCPCH(IS) - UGASCO
C           SPECIFIC HEATS AT IN J/KG/K
            SPCPCH(IS) = SPCPCH(IS)*1000./AMWTCH(IS)
            SPCVCH(IS) = SPCVCH(IS)*1000./AMWTCH(IS)
            SPBSCH(IS) = SPBSCH(IS)*1000./AMWTCH(IS)
10        CONTINUE

          PREBCH(1) = PREFCH(1) - PREECH(1)
          PREBCH(2) = PREFCH(2) - PREECH(2)
C
C         DEBUG PRINT
C
          IF (IDBGCH .NE. 3 .AND. IDBGCH .LT. 1000)  RETURN

          WRITE(JDEBUG,1000)
          WRITE(JDEBUG,1100)
          WRITE(JDEBUG,1200)
          WRITE(JDEBUG,1210) PHI,TEMPMN
          WRITE(JDEBUG,1300)

          DO 20 IS = 1, NSPECH
            GAMMA = SPCPCH(IS)/SPCVCH(IS)
            WRITE(JDEBUG,1400) IS,SPNAME(IS),AMWTCH(IS),SPCPCH(IS),
     1                         SPCVCH(IS),GAMMA,FMHTCH(IS),ENTRCH(IS)
20        CONTINUE

C         -----------------
C         FORMAT STATEMENTS
C         -----------------

1000      FORMAT(//10X,'----------------------' )
1100      FORMAT(  10X,'DEBUG PRINT FROM C2ROCH' )
1200      FORMAT(  10X,'----------------------'/)
```

```
1210    FORMAT (5X,'EQUIVALENCE RATIO = ',E15.6,5X,
     1          5X,'MEAN TEMPERATURE  = ',E15.6/)
1300    FORMAT(1X,'SPECIES',5X,'MOL WT',7X,'CP J/KG/K',6X,'CV J/KG/K',
     1         6X,'GAMMA',10X,'HT FM J/KMOL',3X,'ENTROPY J/KMOL/K'/)
1400    FORMAT(I2,2X,A4,6E15.6)

        RETURN
        END
```

C2SPCA

```
        PROGRAM C2SPCA

        PARAMETER (MSPECH = 14, MNODG2 = 200)
        DIMENSION ATWTCH(MSPECH), N$(2), IOPT$(2)
        DIMENSION TEMP$(MNODG2), CP$(MNODG2), CPLIN$(MNODG2)
        CHARACTER CHARSP*8, SPNAME(MSPECH)*8, YESNO*1, FILNAM*40,
     1            MTITLE*80, PLTITL*96

C*****************************************************************
C
C       THIS PROGRAM CALCULATES THE SPECIFIC HEAT AT CONSTANT PRESSURE
C       FOR SOME SELECTED SPECIES.  THE MOLAL SPECIFIC HEAT IS KNOWN
C       AS FUNCTION OF TEMPERATURE.  THE PROGRAM CAN DETERMINE CP AT
C       A GIVEN TEMPERATURE OR THE MEAN CP FOR A RANGE OF TEMPERATURES.
C       THE PROGRAM ALSO DETERMINES CP AS A LINEAR FUNCTION OF TEMPERA-
C       TURE AND PLOTS THE ACTUAL VARIATION AND THE LINEAR VARIATION
C       VERSUS TEMPERATURE.
C
C*****************************************************************

        JTERMO = 6
        JTERMI = 5
        NSPECH = 0
        JSPOUT = 7
        JOUNT  = 0
        ITOTAL = 0

        OPEN(UNIT=JSPOUT, FILE='SPHEAT.DAT', STATUS='NEW')

C       SET UP THE MOLECULAR WEIGHTS

        ATWTCH(1 ) = 28.013
        ATWTCH(2 ) = 31.999
        ATWTCH(3 ) =  2.016
        ATWTCH(4 ) = 28.01
        ATWTCH(5 ) = 17.008
        ATWTCH(6 ) = 30.006
        ATWTCH(7 ) = 18.015
        ATWTCH(8 ) = 44.01
        ATWTCH(9 ) = 46.006
        ATWTCH(10) = 16.043
        ATWTCH(11) = 28.053
```

```
      ATWTCH(12) = 30.069
      ATWTCH(13) = 44.096
      ATWTCH(14) = 58.122

C     SET UP THE NAMES OF THE MOLECULES

      SPNAME(1 ) = 'N2      '
      SPNAME(2 ) = 'O2      '
      SPNAME(3 ) = 'H2      '
      SPNAME(4 ) = 'CO      '
      SPNAME(5 ) = 'OH      '
      SPNAME(6 ) = 'NO      '
      SPNAME(7 ) = 'H2O     '
      SPNAME(8 ) = 'CO2     '
      SPNAME(9 ) = 'NO2     '
      SPNAME(10) = 'CH4     '
      SPNAME(11) = 'C2H4    '
      SPNAME(12) = 'C2H6    '
      SPNAME(13) = 'C3H8    '
      SPNAME(14) = 'C4H10   '

C     UNIVERSAL GAS CONSTANT IN KJ/KMOL/K

      UGASCO = 8.31434

      WRITE (JTERMO,1000)
1000  FORMAT(' INPUT INITIAL, INCREMENTAL AND FINAL TEMPERATURES'/
     1        ' ===> ',$)
      READ  (JTERMI,*) TINIT,TINCR,TFINAL

      WRITE (JTERMO,1001)
1001  FORMAT(
     1 ' DO YOU WANT TO HAVE ALL VALUES WRITTEN IN FILES [Y/N]')
      READ(JTERMI,1800) YESNO
      IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') ITOTAL = 1

      MTITLE = 'SPECIFIC HEAT VERSUS TEMPERATURE'
      CALL GR_INIT(JTERMI,JTERMO,MTITLE)

C     READ THE NAME OF THE SPECIES

10    WRITE (JTERMO,1100)
1100  FORMAT(' INPUT THE NAME OF THE SPECIES'/' ==> ',$)
      CHARSP = ' '
      READ  (JTERMI,1200) CHARSP
1200  FORMAT(A8)

      DO 20 I = 1, MSPECH
        IF (CHARSP .EQ. SPNAME(I)) THEN
          ITYPE = I
          GOTO 30
        ENDIF
20    CONTINUE

      WRITE (JTERMO,1300) CHARSP
1300  FORMAT(' THE SPECIES NAME IS NOT FOUND IN THE LIST :  ',A8)
```

```
      STOP

30    KOUNT  = 0
      TEMP   = TINIT
      NSPECH = NSPECH + 1
      SUMCP  = 0.
      SUMCPT = 0.
      SUMTM  = 0.
      SUMTN  = 0.
      SUMTM2 = 0.

      WRITE (JSPOUT,1900) CHARSP
1900  FORMAT(//3X,'SPECIFIC HEATS FOR : ',A8/)
      WRITE (JSPOUT,1400)
1400  FORMAT(//5X,'TEMPERATURE K',2X'CP KJ/KMOL/K',3X,'CV KJ/KMOL/K',
     1          3X,'CP KJ/KG/K',5X,'CV KJ/KG/K',5X,'GAMMA'/)

40    THE   = 0.01*TEMP
      CP    = CPSP(ITYPE,THE)
      THEN  = 0.01*THE
      THEN2 = THEN*THEN
      CPT   = CP*THEN
      CV    = CP - UGASCO
      CPU   = CP/ATWTCH(ITYPE)
      CVU   = CV/ATWTCH(ITYPE)
      GAMMA = CP/CV
      WRITE (JSPOUT,1500) TEMP,CP,CV,CPU,CVU,GAMMA
1500  FORMAT(2X,6E15.6)
      KOUNT = KOUNT + 1
      SUMCP = SUMCP + CP
      SUMTM = SUMTM + TEMP
      SUMTN = SUMTN + THEN
      SUMCPT = SUMCPT + CPT
      SUMTM2 = SUMTM2 + THEN2
      TEMP$(KOUNT) = TEMP
      CP$(KOUNT) = CP
      TEMP  = TEMP+TINCR
      IF (TEMP .LE. TFINAL) GOTO 40
      TMAV  = SUMTM/KOUNT
      CPAV  = SUMCP/KOUNT
      CVAV  = CPAV - UGASCO
      GAMMA = CPAV/CVAV
      CPU   = CPAV/ATWTCH(ITYPE)
      CVU   = CVAV/ATWTCH(ITYPE)
      ANUE  = SUMTN*SUMCP - KOUNT*SUMCPT
      DENO  = SUMTN*SUMTN - KOUNT*SUMTM2
      SLOPE = ANUE/DENO
      TCEPT = (SUMCP-SLOPE*SUMTN)/FLOAT(KOUNT)
      SLOPE = SLOPE*0.0001
      WRITE (JSPOUT,1600)
1600  FORMAT (//3X,'AVERAGE VALUES')
      WRITE (JSPOUT,1500) TMAV,CPAV,CVAV,CPU,CVU,GAMMA
      WRITE (JSPOUT,1550) CHARSP, TCEPT, SLOPE
1550  FORMAT(2X,'CP (',A8,' ) = ',G14.5,'  + T ',G14.5)

      DO 50 IK = 1, KOUNT
        TEMP$(IK+KOUNT) = TEMP$(IK)
```

```
          CP$  (IK+KOUNT) = TCEPT + SLOPE*TEMP$(IK)
          CPLIN$(IK)      = TCEPT + SLOPE*TEMP$(IK)
50      CONTINUE

        IOPT$(1) = 4
        IOPT$(2) = 2
        N$(1)    = KOUNT
        N$(2)    = KOUNT
        PLTITL   = ' '
        NLINE    = 2
        INDGR    = 21
        WRITE(JTERMO,1650)
1650    FORMAT(' WANT TO PLOT CURRENT SPECIES DATA [Y/N] ?'/' ==> ',$)
        READ(JTERMI,1800) YESNO

        IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y')
     1     CALL GR_LINE(IOPT$,NLINE,PLTITL,INDGR,TEMP$,CP$,N$)

        IF (ITOTAL .EQ. 0) GOTO 2345
        FILNAM=' '
        FILNAM(1:5)='S.DAT'
        JOUNT=JOUNT+1
        WRITE(FILNAM(6:6),1778) JOUNT
1778    FORMAT(I1)
        OPEN (UNIT=48,FILE=FILNAM,STATUS='NEW')
        WRITE(48,*) KOUNT
        DO 1134 I = 1, KOUNT
           WRITE(48,1005) TEMP$(I),CP$(I),CPLIN$(I)
1134    CONTINUE

1005    FORMAT(3E15.7)
        CLOSE(48)

2345    WRITE(JTERMO,1700)
1700    FORMAT(' MORE SPECIES [Y/N] ?'/' ==> ',$)
        READ(JTERMI,1800) YESNO
1800    FORMAT(A1)

        IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 10
C
        STOP
        END
```

CHKBN2

```
        SUBROUTINE CHKBN2 (LCELL,MEM1,MEM2,MEM3,MEM4,NERR,NAME)
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        CHARACTER NAME*6, ERRTYP*30
C
C*****************************************************************
```

```
C
C       THIS SUBROUTINE CHECKS THE ASSIGNMENTS OF THE BOUNDARY NODE
C       ARRAY.  THESE ASSIGNMENTS ARE PRONE TO ERROR AFTER THE GRID-
C       DIVIDE AND GRID-COLLAPSE ROUTINES.  HENCE THIS ROUTINE MUST BE
C       USED AS A DEBUG CHECK AFTER CALLS TO THESE GRID CHANGING ROUTINES
C       IS MADE. LCELL IS THE MOST RECENTLY DIVIDED OR COLLAPSED CELL;
C       WHERE THE ERROR MIGHT OCCUR.  MEM1 THRU MEM4 ARE THE SUBCELLS OF
C       LCELL IF IT WERE COLLAPSED.  NAME INDICATES WHEN AND WHERE
C       THE ERROR OCCURED.  NERR COUNTS NUMBER OF ERRORS.
C
C***********************************************************************
C
C       COUNT THE PREVIOUS NUMBER OF ERRORS AND SET DEBUG UNIT
C
        NERRP  = NERR
        OPEN (UNIT=JDUMY2, FILE='CHKBN2.DAT', STATUS='NEW')
C
        DO 150 IBOUND = 1, NBNDG2
C
          INODE  = IBNDG2(1,IBOUND)
          IONE   = IBNDG2(2,IBOUND)
          ITWO   = IBNDG2(3,IBOUND)
          IEDGE  = IBNDG2(4,IBOUND)
C
C         CHECK IF THE BOUNDARY NODE IS MARKED FOR DELETE
C
          IF (INODE .EQ. -9) GO TO 150
C
C         CHECK OUT THE CELLS ADJACENT TO THE BOUNDARY NODES AND
C         ASSIGNMENT OF THE NODE ITSELF
C
          GO TO (105,110,115,120,125,130,135,140), (IEDGE-1)
          GO TO 150
C
C2        SOUTHWESTERN CORNER
105       ICONE  = NEIBG2(3,INODE)
          ICTWO  = 0
          ICNODE = ICELG2(2,ICONE)
          GO TO 145
C
C3        SOUTHERN EDGE
110       ICONE  = NEIBG2(4,INODE)
          ICTWO  = NEIBG2(3,INODE)
          ICNODE = ICELG2(4,ICONE)
          GO TO 145
C
C4        SOUTHEASTERN CORNER
115       ICONE  = NEIBG2(4,INODE)
          ICTWO  = 0
          ICNODE = ICELG2(4,ICONE)
          GO TO 145
C
C5        EASTERN EDGE
120       ICONE  = NEIBG2(1,INODE)
          ICTWO  = NEIBG2(4,INODE)
          ICNODE = ICELG2(6,ICONE)
          GO TO 145
```

```
C
C6        NORTHEASTERN CORNER
125       ICONE  = NEIBG2(1,INODE)
          ICTWO  = 0
          ICNODE = ICELG2(6,ICONE)
          GO TO 145
C
C7        NORTHERN EDGE
130       ICONE  = NEIBG2(2,INODE)
          ICTWO  = NEIBG2(1,INODE)
          ICNODE = ICELG2(8,ICONE)
          GO TO 145
C
C8        NORTHWESTERN CORNER
135       ICONE  = NEIBG2(2,INODE)
          ICTWO  = 0
          ICNODE = ICELG2(8,ICONE)
          GO TO 145
C
C9        WESTERN EDGE
140       ICONE  = NEIBG2(3,INODE)
          ICTWO  = NEIBG2(2,INODE)
          ICNODE = ICELG2(2,ICONE)
C
145       IF (IONE .NE. ICONE) THEN
            ERRTYP = 'ERROR IN FIRST ADJACENT CELL  '
            NERR = NERR + 1
            WRITE(JDUMY2,1000) IBOUND, IEDGE , INODE , IONE , ITWO ,
     1                                 ERRTYP, ICNODE, ICONE, ICTWO
          ENDIF
C
          IF (ITWO .NE. ICTWO) THEN
            ERRTYP = 'ERROR IN SECOND ADJACENT CELL '
            NERR = NERR + 1
            WRITE(JDUMY2,1000) IBOUND, IEDGE , INODE , IONE , ITWO ,
     1                                 ERRTYP, ICNODE, ICONE, ICTWO
          ENDIF
C
          IF (INODE.NE. ICNODE) THEN
            ERRTYP = 'ERROR IN NODE ASSIGNMENT      '
            NERR = NERR + 1
            WRITE(JDUMY2,1000) IBOUND, IEDGE , INODE , IONE , ITWO ,
     1                                 ERRTYP, ICNODE, ICONE, ICTWO
          ENDIF
C
C         GO BACK FOR NEXT BOUNDARY NODE
C
150     CONTINUE
C
        IF (NERR .NE. NERRP) THEN
           WRITE(JTERMO,1100) NAME,NITRE2
           WRITE(JDUMY2,1100) NAME,NITRE2
           WRITE(JDUMY2,1200) LCELL, MEM1, MEM2, MEM3, MEM4
           CLOSE(UNIT=JDUMY2, DISP='KEEP')
        ELSE
           CLOSE(UNIT=JDUMY2, DISP='DELETE')
        ENDIF
```

```
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(2X,'IBOUND =',I5,5X,'IEDGE  =',I5,
     1         5X,'INODE  =',I5,5X,'IONE   =',I5,5X,'ITWO  =',I5/2X,
     2     A30,6X,'ICNODE =',I5,5X,'ICONEL =',I5,5X,'ICTWO =',I5)
1100    FORMAT(2X,'ERROR ',A6,5X,'AFTER',I5,2X,'ITERATIONS IN CHKBN2'/)
1200    FORMAT(2X,'LCELL =',I5,5X,'MEM1 =',I5,5X,'MEM2 =',I5,
     1                          5X,'MEM3 =',I5,5X,'MEM4 =',I5/)

        RETURN
        END
```

CHKMAS

```
        SUBROUTINE CHKMAS

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'
C
C***********************************************************************

C       THIS SUBROUTINE CALCULATES THE MASS FLOW RATE AT A VERTICAL
C       PLANE STARTING FROM A GIVEN NODE AT THE BOTTOM OF THE PLANE.

C***********************************************************************
C
C       READ THE FOLLOWING FUEL QUANTITIES
C          IBASE   : THE BASE NODE OF THE PLANE OF INJECTION
C
        READ (JREADS,*) IBASE
        INODE = IBASE
        NBTYPE = 0
        NB1    = NEIBG2(4,INODE)
        NB2    = NEIBG2(3,INODE)

        IF (NB1 .NE. 0) THEN
           NBTYPE = 4
           INTYPE = 6
        ELSEIF (NB2 .NE. 0) THEN
           NBTYPE = 3
           INTYPE = 8
        ENDIF
C
C       ERROR CONDITION
C
        IF (NBTYPE .EQ. 0) THEN
           ZER1 = ISTART
           ZER2 = NBTYPE
```

```
          CALL ERRORM (46,'H2SCRI','ISTART',ZER1,'NBTYPE',ZER2,JPRINT,
     1             'ERROR IN NEIGHBOUR CELLS OF STARTING POINT')
        ENDIF
C
C       NOW MARCH IN THE APPROPRIATE DIRECTION
C
        KOUNT  = 0
        SUMMAS = 0.

10      KOUNT  = KOUNT + 1
C       FIND THE NEXT CELL ON TOP OF THE NODE UNDER CONSIDERATION
        NBNEXT = NEIBG2(NBTYPE,INODE)
C       SEE IF YOU HAVE REACHED THE TOP BOUNDARY SURFACE
        IF (NBNEXT .EQ. 0) GOTO 20
C       CALCULATE THE DENSITY, VELOCITY AND Y-DISTANCE AT THE LOWER NODE
        RHOL   = DPENG2(1,INODE)
        UL     = DPENG2(2,INODE)/DPENG2(1,INODE)
        YL     = GEOMG2(2,INODE)
C       FIND THE UPPER NODE AND THE CORRESPONDING QUANTITIES
        INODE  = ICELG2(INTYPE,NBNEXT)
        RHOU   = DPENG2(1,INODE)
        UU     = DPENG2(2,INODE)/DPENG2(1,INODE)
        YU     = GEOMG2(2,INODE)
C       COMPUTE AVERAGE DENSITY AND VELOCITY
        RHO    = 0.5*(RHOL+RHOU)
        U      = 0.5*(UL+UU)
C       SUM THE MASS FLOW RATE FOR THIS CELL
        SUMMAS = SUMMAS + RHO*U*(YU-YL)
        GO TO 10

20      CONTINUE
C       WRITE ALL THE OUTPUT
C
        WRITE (JTERMO,*) ' ****** WRITTING OUTPUT ON CHKMAS.DAT ******'
        OPEN (UNIT=JDUMY1, FILE='CHKMAS.DAT', STATUS='NEW')
        WRITE (JDUMY1,30) KOUNT, SUMMAS
30      FORMAT (5X,'TOTAL NODES IN THE PLANE:',I4,5X,
     1            ' MASS FLOW RATE',G14.5)

        RETURN
        END
```

CHKNC2

```
        SUBROUTINE CHKNC2 (LCELL,MEM1,MEM2,MEM3,MEM4,NERR,NAME)
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        CHARACTER NAME*6
C
```

```
C***********************************************************************
C
C       THIS SUBROUTINE CHECKS THE NEIGHBOURS OF THE CORNER NODES OF ALL
C       CELLS TAKING ONE CALL AT A TIME.  THESE ASSIGNMENTS ARE PRONE TO
C       ERROR AFTER THE GRID-DIVIDE AND GRID-COLLAPSE ROUTINES.  HENCE
C       THIS ROUTINE MUST BE USED AS A DEBUG CHECK AFTER CALLS TO GRID
C       CHANGING ROUTINES IS MADE.  LCELL IS THE MOST RECENTLY DIVIDED
C       OR COLLAPSED CELL; WHERE THE ERROR MIGHT OCCUR.  MEM1 THRU MEM4
C       ARE THE SUBCELLS OF LCELL IF IT WERE COLLAPSED.  NAME INDICATES
C       WHEN AND WHERE THE ERROR OCCURED.  NERR COUNTS NUMBER OF ERRORS.
C
C
C***********************************************************************
C
C       COUNT THE PREVIOUS NUMBER OF ERRORS AND SET DEBUG UNIT
C
        NERRP  = NERR
        OPEN (UNIT=JDUMY2, FILE='CHKNC2.DAT', STATUS='NEW')
C
C       THERE IS NO NEED TO CHECK THE NON-CEWIC CELLS SINCE THE
C       NEIGHBOUR-CELL ARRAY ONLY POINTS TO THE FINE CELLS

        DO 10 ICL = 1, NCELG2
          KC    = ICELG2(1,ICL)
          IF (KC .NE. 0) GO TO 10

C         COMPUTE THE CORNER NODES

          KSW    = ICELG2(2,ICL)
          KSE    = ICELG2(4,ICL)
          KNE    = ICELG2(6,ICL)
          KNW    = ICELG2(8,ICL)

C         FIND THE NEIGHBOURS (CELLS) OF THESE NODES POINTING INWARD

          NB1   = NEIBG2(3,KSW)
          NB2   = NEIBG2(4,KSE)
          NB3   = NEIBG2(1,KNE)
          NB4   = NEIBG2(2,KNW)

C         CHECK THE SOUTHWEST-NODE-NEIGHBOUR

          IF (NB1 .NE. ICL) THEN
             NERR  = NERR + 1
             NTP = 1
             WRITE(JDUMY2,1000) ICL,KSW,KSE,KNE,KNW,NTP,NB1,NB2,NB3,NB4
          ENDIF

C         CHECK THE SOUTHEAST-NODE-NEIGHBOUR

          IF (NB2 .NE. ICL) THEN
             NERR  = NERR + 1
             NTP = 2
             WRITE(JDUMY2,1000) ICL,KSW,KSE,KNE,KNW,NTP,NB1,NB2,NB3,NB4
          ENDIF

C         CHECK THE NORTHEAST-NODE-NEIGHBOUR
```

```
          IF (NB3 .NE. ICL) THEN
             NERR  = NERR + 1
             NTP = 3
             WRITE(JDUMY2,1000) ICL,KSW,KSE,KNE,KNW,NTP,NB1,NB2,NB3,NB4
          ENDIF

C         CHECK THE NORTHWEST-NODE-NEIGHBOUR

          IF (NB4 .NE. ICL) THEN
             NERR  = NERR + 1
             NTP = 4
             WRITE(JDUMY2,1000) ICL,KSW,KSE,KNE,KNW,NTP,NB1,NB2,NB3,NB4
          ENDIF
C
C         GO BACK FOR NEXT CELL
C
10      CONTINUE
C
        IF (NERR .NE. NERRP) THEN
           WRITE(JTERMO,1100) NAME,NITRE2
           WRITE(JDUMY2,1100) NAME,NITRE2
           WRITE(JDUMY2,1200) LCELL, MEM1, MEM2, MEM3, MEM4
           JPRINT = JDUMY3
           CALL G2PRNT(15)
           CLOSE(UNIT=JDUMY2, DISP='KEEP')
        ELSE
           CLOSE(UNIT=JDUMY2, DISP='DELETE')
        ENDIF
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(2X,'ICL =',I5,5X,'KSW =',I5,5X,'KSE =',I5,5X,'KNE =',I5,
     1         5X,'KNW =',I5/5X,'NTP =',I5,5X,'NB1 =',I5,5X,'NB2 =',I5,
     2         5X,'NB3 =',I5,5X,'NB4 =',I5                              )
1100    FORMAT(2X,'ERROR ',A6,5X,'AFTER',I5,2X,'ITERATIONS IN CHKNC2'/)
1200    FORMAT(2X,'LCELL =',I5,5X,'MEM1 =',I5,5X,'MEM2 =',I5,
     1                          5X,'MEM3 =',I5,5X,'MEM4 =',I5/)

        RETURN
        END
```

CHKNN2

```
        SUBROUTINE CHKNN2 (LCELL,MEM1,MEM2,MEM3,MEM4,NERR,NAME)
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
```

```
      CHARACTER NAME*6, ERRTYP*30
      LOGICAL CHECK1, CHECK2, CHECK3, CHECK4, CHECKA
C
C*********************************************************************
C
C     THIS SUBROUTINE CHECKS THE NEIGHBOURS OF ALL THE NODES TAKING
C     ONE NODE AT A TIME.  THESE ASSIGNMENTS ARE PRONE TO ERROR AFTER
C     THE GRID-DIVIDE AND GRID-COLLAPSE ROUTINES.  HENCE THIS ROUTINE
C     MUST BE USED AS A DEBUG CHECK AFTER CALLS TO GRID CHANGING
C     ROUTINES IS MADE.  LCELL IS THE MOST RECENTLY DIVIDED OR
C     COLLAPSED CELL; WHERE THE ERROR MIGHT OCCUR.  MEM1 THRU MEM4
C     ARE THE SUBCELLS OF LCELL IF IT WERE COLLAPSED.  NAME INDICATES
C     WHEN AND WHERE THE ERROR OCCURED.  NERR COUNTS NUMBER OF ERRORS.
C
C*********************************************************************
C
C     COUNT THE PREVIOUS NUMBER OF ERRORS AND SET DEBUG UNIT
C
      NERRP  = NERR
      OPEN (UNIT=JDUMY2, FILE='CHKNN2.DAT', STATUS='NEW')
C
C     STEP OVER ALL THE NODES TO CHECK NEIGHBOURS
C
      DO 30 INODE = 1, NNODG2
C
C       CHECK IF THE NODE IS MARKED FOR COLLAPSE
C
        IF (DPENG2(1,INODE) .EQ. -99.) GO TO 30
C
C       COMPUTE THE NEIGHBOUR CELLS
C
        NBSW    = NEIBG2(1,INODE)
        NBSE    = NEIBG2(2,INODE)
        NBNE    = NEIBG2(3,INODE)
        NBNW    = NEIBG2(4,INODE)
C
C       CHECK IF THE NEIGHBOUR CELLS ARE NOT WITHIN BOUNDS
C
        CHECK1 = NBSW .LT. 0  .OR.  NBSW .GT. NCELG2
        CHECK2 = NBSE .LT. 0  .OR.  NBSE .GT. NCELG2
        CHECK3 = NBNE .LT. 0  .OR.  NBNE .GT. NCELG2
        CHECK4 = NBNW .LT. 0  .OR.  NBNW .GT. NCELG2
        CHECKA = CHECK1 .OR. CHECK2 .OR. CHECK3 .OR. CHECK4
C
        IF (CHECKA) THEN
          NERR   =  NERR + 1
          ERRTYP = 'OUT OF BOUND NEIGHBOUR CELL   '
          WRITE(JDUMY2,1000) ERRTYP, INODE, NBSW, NBSE, NBNE, NBNW
          GOTO 30
        ENDIF
C
C       CHECK IF ALL THE NEIGHBOUR CELLS ARE UNDEFINED
C
        CHECK1 = NBSW .EQ. 0
        CHECK2 = NBSE .EQ. 0
        CHECK3 = NBNE .EQ. 0
        CHECK4 = NBNW .EQ. 0
```

```
      CHECKA = CHECK1 .AND. CHECK2 .AND. CHECK3 .AND. CHECK4
C
      IF (CHECKA) THEN
         NERR   =  NERR + 1
         ERRTYP = 'ALL NEIGHBOUR CELLS UNDEFINED '
         WRITE(JDUMY2,1000) ERRTYP, INODE, NBSW, NBSE, NBNE, NBNW
         GOTO 30
      ENDIF
C
C     IF SOME OF THE NEIGHBOUR CELLS ARE ZERO THEN THE NODE
C     MUST BE A BOUNDARY NODE
C
      CHECKA = CHECK1 .OR. CHECK2 .OR. CHECK3 .OR. CHECK4
C
      IF (CHECKA) THEN
         DO 10 INBND = 1, NBNDG2
            IF (IBNDG2(1,INBND) .EQ. INODE) GOTO 20
10       CONTINUE
c        this node is not a boundary node; however, it may be a
c        corner node of an internal boundary with three non-zero
c        neighbour cells
         inboun = 0
         if (check1) inboun = inboun + 1
         if (check2) inboun = inboun + 1
         if (check3) inboun = inboun + 1
         if (check4) inboun = inboun + 1
         if (inboun .le. 1) goto 20
         NERR   =  NERR + 1
         ERRTYP = 'INTERIOR PT HAS ZERO NEIGHBOUR'
         WRITE(JDUMY2,1000) ERRTYP, INODE, NBSW, NBSE, NBNE, NBNW
      ENDIF
C
20    IF (NBSW .EQ. 0) THEN
         NODE1 = INODE
      ELSE
         NODE1 = ICELG2(6,NBSW)
         IF (NBSW .EQ. NBNW) NODE1 = ICELG2(5,NBSW)
         IF (NBSW .EQ. NBSE) NODE1 = ICELG2(7,NBSW)
      ENDIF
C
      IF (NBSE .EQ. 0) THEN
         NODE2 = INODE
      ELSE
         NODE2 = ICELG2(8,NBSE)
         IF (NBSW .EQ. NBSE) NODE2 = ICELG2(7,NBSW)
         IF (NBSE .EQ. NBNE) NODE2 = ICELG2(9,NBSE)
      ENDIF
C
      IF (NBNE .EQ. 0) THEN
         NODE3 = INODE
      ELSE
         NODE3 = ICELG2(2,NBNE)
         IF (NBSE .EQ. NBNE) NODE3 = ICELG2(9,NBSE)
         IF (NBNE .EQ. NBNW) NODE3 = ICELG2(3,NBNE)
      ENDIF
C
      IF (NBNW .EQ. 0) THEN
```

```
            NODE4 = INODE
          ELSE
            NODE4 = ICELG2(4,NBNW)
            IF (NBSW .EQ. NBNW) NODE4 = ICELG2(5,NBSW)
            IF (NBNE .EQ. NBNW) NODE4 = ICELG2(3,NBNE)
          ENDIF
C
C         CHECK IF ALL THE NEIGHBOUR CELLS AGREE ON NODE ASSIGNMENTS
C
          CHECK1 = NODE1 .NE. INODE
          CHECK2 = NODE2 .NE. INODE
          CHECK3 = NODE3 .NE. INODE
          CHECK4 = NODE4 .NE. INODE
          CHECKA = CHECK1 .OR. CHECK2 .OR. CHECK3 .OR. CHECK4
C
          IF (CHECKA) THEN
            NERR   =  NERR + 1
            ERRTYP = 'NODE ASSIGNMENT ERROR          '
            WRITE(JDUMY2,1100) ERRTYP, INODE, NBSW , NBSE , NBNE ,
     1                         NBNW  , NODE1, NODE2, NODE3, NODE4
          ENDIF
C
C         GO BACK FOR NEXT NODE
C
30      CONTINUE
C
        IF (NERR .NE. NERRP) THEN
          WRITE(JTERMO,1200) NAME,NITRE2
          WRITE(JDUMY2,1200) NAME,NITRE2
          WRITE(JDUMY2,1300) LCELL, MEM1, MEM2, MEM3, MEM4
          CLOSE(UNIT=JDUMY2, DISP='KEEP')
        ELSE
          CLOSE(UNIT=JDUMY2, DISP='DELETE')
        ENDIF
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(2X,A30,2X,'INODE =',I5,5X,'NBSW  =',I5,5X,'NBSE  =',I5,
     1                              5X,'NBNE  =',I5,5X,'NBNW  =',I5)
1100    FORMAT(2X,A30,2X,'INODE =',I5,5X,'NBSW  =',I5,5X,'NBSE  =',I5,
     1       5X,'NBNE  =',I5,5X,'NBNW  =',I5/34X,'NODE1 =',I5,5X,
     2          'NODE2 =',I5,5X,'NODE3 =',I5, 5X,'NODE4 =',I5)
1200    FORMAT(2X,'ERROR ',A6,5X,'AFTER',I5,2X,'ITERATIONS IN CHKNN2'/)
1300    FORMAT(2X,'LCELL =',I5,5X,'MEM1 =',I5,5X,'MEM2 =',I5,
     1                         5X,'MEM3 =',I5,5X,'MEM4 =',I5/)
C
        RETURN
        END
```

CHKPR2

```
      SUBROUTINE CHKPR2 (INODE)

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'PRCOMN.INC'
      DIMENSION DPLEFT(MEQNFL), DPRITE(MEQNFL)

C*********************************************************************

C     THIS SUBROUTINE CORRECTS THE CONSERVATIVE VARIABLES AT A GIVEN
C     NODE 'INODE', IF THE PRESSURE AT THAT NODE BECOMES NEGATIVE.
C     IT IS HOPED THAT SUCH A SITUATION ONLY OCCURS AT A FEW NODES.
C     IF THIS OCCURS IN A REGION WITH MORE THAN ONE NODE THAN THE
C     CALCULATION WILL BECOME UNSTABLE ANYWAY.  SO THIS SUBROUTINE
C     IS ACTUALLY FAIL SAFE.

C*********************************************************************

      WRITE(JDEBUG,*) ' NODE WITH NEG PR =',INODE
      NB1 = NEIBG2(1,INODE)
      NB2 = NEIBG2(2,INODE)
      NB3 = NEIBG2(3,INODE)
      NB4 = NEIBG2(4,INODE)

C     THE CORRECTION IS NOT APPLIED AT THE CORNER BOUNDARY CELLS

      IF (NB1 .EQ. 0 .AND. NB4 .EQ. 0) RETURN
      IF (NB2 .EQ. 0 .AND. NB3 .EQ. 0) RETURN

C     SETUP THE LEFT NEIGHBOUR CELL AND ITS NODE POINTER

      IF (NB1 .NE. 0) THEN
        NBLEFT = NB1
        IPLEFT = 8
      ELSE
        NBLEFT = NB4
        IPLEFT = 2
      ENDIF
C
C     FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
      IF (NB1 .EQ. NB4) THEN
        IPLEFT = 9
        IF (ICELG2(9,NBLEFT) .EQ. 0) THEN
          INLFT1 = ICELG2(2,NBLEFT)
          INLFT2 = ICELG2(8,NBLEFT)
          XLEFT  = 0.5*(GEOMG2(1,INLFT1)+GEOMG2(1,INLFT2))
          YLEFT  = 0.5*(GEOMG2(2,INLFT1)+GEOMG2(2,INLFT2))
          DO 10 IQ = 1, NEQNFL
```

```
            DPLEFT(IQ) = 0.5*(DPENG2(IQ,INLFT1)+DPENG2(IQ,INLFT2))
10          CONTINUE
            GOTO 30
          ENDIF
        ENDIF

C       COMPUTE THE LEFT NODE, DISTANCES AND DP VARIABLES

        INLEFT = ICELG2(IPLEFT,NBLEFT)

        XLEFT  = GEOMG2(1,INLEFT)
        YLEFT  = GEOMG2(2,INLEFT)
        DO 20 IQ = 1, NEQNFL
           DPLEFT(IQ) = DPENG2(IQ,INLEFT)
20      CONTINUE

30      CONTINUE

C       SETUP THE RIGHT NEIGHBOUR CELL AND ITS NODE POINTER

        IF (NB2 .NE. 0) THEN
          NBRITE = NB2
          IPRITE = 6
        ELSE
          NBRITE = NB3
          IPRITE = 4
        ENDIF
C
C       FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
        IF (NB2 .EQ. NB3) THEN
          IPRITE = 5
          IF (ICELG2(5,NBRITE) .EQ. 0) THEN
            INRIT1 = ICELG2(4,NBRITE)
            INRIT2 = ICELG2(6,NBRITE)
            XRITE  = 0.5*(GEOMG2(1,INRIT1)+GEOMG2(1,INRIT2))
            YRITE  = 0.5*(GEOMG2(2,INRIT1)+GEOMG2(2,INRIT2))
            DO 40 IQ = 1, NEQNFL
              DPRITE(IQ) = 0.5*(DPENG2(IQ,INRIT1)+DPENG2(IQ,INRIT2))
40          CONTINUE
            GOTO 60
          ENDIF
        ENDIF

C       COMPUTE THE RIGHT NODE, DISTANCES AND DP VARIABLES

        INRITE = ICELG2(IPRITE,NBRITE)
        XRITE  = GEOMG2(1,INRITE)
        YRITE  = GEOMG2(2,INRITE)

        DO 50 IQ = 1, NEQNFL
           DPRITE(IQ) = DPENG2(IQ,INRITE)
50      CONTINUE

60      CONTINUE

        XNODE  = GEOMG2(1,INODE)
```

```
      YNODE  = GEOMG2(2,INODE)
      SNODE2 = (XNODE-XLEFT)**2 + (YNODE-YLEFT)**2
      SRITE2 = (XRITE-XLEFT)**2 + (YRITE-YLEFT)**2
      RATIO  = SQRT(SNODE2/SRITE2)
C
C     DO THE INTERPOLATION
C
      DO 70 IQ = 1, NEQNFL
c        DPENG2(IQ,INODE) = DPLEFT(IQ) +
c    1     ( DPRITE(IQ) - DPLEFT(IQ) )*RATIO
         DPHERE = DPLEFT(IQ) + (DPRITE(IQ) -DPLEFT(IQ))*RATIO
         DPENG2(IQ,INODE) = 0.5*(DPHERE + DPENG2(IQ,INODE))
70    CONTINUE
C
C     NOW RECOMPUTE THE PRESSURE, TEMPERATURE ETC.
C
      RHORPR = DPENG2(1,INODE)
      UCOMPR = DPENG2(2,INODE)/RHORPR
      VCOMPR = DPENG2(3,INODE)/RHORPR
      BEPSPR = DPENG2(4,INODE)
      BE     = BEPSPR/RHORPR
      VELO2U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BE
      VELO2  = FMREFL*VELO2U

C     COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

      SUMY   = 0.
      YUPPER = 1. - YNRTCH

      DO 80 IS = 1, NEQSCH

          JS         = NEQBAS + IS
          YSPEPR(IS) = DPENG2(JS,INODE)/DPENG2(1,INODE)

          IF (YSPEPR(IS) .LT. 0.) THEN
            YSPEPR(IS)       = 0.
            DPENG2(JS,INODE) = 0.
          ENDIF

          IF (YSPEPR(IS) .GT. YUPPER) THEN
            YSPEPR(IS)       = YUPPER
            DPENG2(JS,INODE) = YUPPER*DPENG2(1,INODE)
          ENDIF

          SUMY       = SUMY + YSPEPR(IS)

80    CONTINUE

      YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
C     YSPEPR(NEQSCH+1) = ABS(1. - SUMY - YNRTCH)
      IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.

      SYSHFS = 0.
```

```
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
C     COMPUTE THE TEMPERATURE IN DEGREE K
C
      DO 90 IS = 1, NSPECH
          SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
          SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
          SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
          BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
90    CONTINUE

      BIGBM   = SYSCPS - UGASFL*SYSBMS
      BIGCM   = BE - 0.5*VEL02 - SYSHFS + TREFCH*SYSCPS
     1                + 0.5*TREFCH*TREFCH*BIGAM
      IF (BIGAM .LT. 1.E-10) THEN
         TEMPPR = BIGCM/BIGBM
      ELSE
         DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
         TEMPPR = ( SQRT(DISCRI)-BIGBM )/BIGAM
      ENDIF
C
C     NORMALIZE THE TEMPERATURE
C
      TEMPPR = TEMPPR/TREFFL
C
C     COMPUTE THE DIMENSIONLESS PRESSURE
C
      PRESPR  = RHORPR*TEMPPR*AMWTFL*SYSBMS
C
C     SAVE THE PRESSURE AND TEMPERATURE AT THE NODE
C
      PRESG2(INODE) = PRESPR
      TEMPG2(INODE) = TEMPPR

      RETURN
      END
```

CHKREF

```
      SUBROUTINE CHKREF
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
      INCLUDE 'PRCOMN.INC'
```

```
C
C*********************************************************************
C
C       THIS SUBROUTINE RE-INITIALIZES THE REFERENCE VALUES FOR DENSITY,
C       TEMPERATURE, OR PRESSURE.  IT ASSUMES THAT THE NON-DIMENSIONAL
C       DENSITY, PRESSURE, TEMPERATURE, AND VELOCITY REMAINS CONSTANT AND
C       REASSIGNS THE VALUES OF THE ENERGY (EPSILON) FOR THE WHOLE DOMAIN.

C*********************************************************************
C
C       NOTE THAT FOR THIS CASE ONLY REFERENCE TEMPERATURE IS ALLOWED TO
C       CHANGE
C
        write(6,*) ' old rhorfl',rhorfl,TREFFL,UREFFL

        RHORFL = RHORFL*TREFFL/APASKY(7)*APASKY(9)/PRESFL
        TREFFL = APASKY(7)
        PRESFL = APASKY(9)
        UREFFL = SQRT(PRESFL/RHORFL)

        write(6,*) ' new rhorfl',rhorfl,UREFFL
        FMREFL = UREFFL**2
        WDREFL = RHORFL*UREFFL/DISTFL

        DO 30 INODE = 1, NNODG2

          RHORPR = DPENG2(1,INODE)
          UCOMPR = DPENG2(2,INODE)/RHORPR
          VCOMPR = DPENG2(3,INODE)/RHORPR
          VEL02U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C         COMPUTE THE DIMENSIONAL QUANTITIES
C
          UCOMPD = UCOMPR*UREFFL
          VCOMPD = VCOMPR*UREFFL
          RHOD   = RHORPR*RHORFL

C         COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

          SUMY   = 0.
          DO 10 IS = 1, NEQSCH
             JS         = NEQBAS + IS
             YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
             SUMY       = SUMY + YSPEPR(IS)
10        CONTINUE

          YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
C
C         COMPUTE SOME DIMENSIONAL NUMBERS
C
          SYSHFS = 0.
          SYSCPS = 0.
          SYSBMS = 0.
          BIGAM  = 0.
C
          DO 20 IS = 1, NSPECH
             SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
```

```
            SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
            SYSBMS = SYSBMS + YSPEPR(IS)/AMWTCH(IS)
            BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
20        CONTINUE
C
C         COMPUTE THE TEMPERATURE IN DEGREE K AND PRESSURE IN PA
C
          TEMPD  = TEMPG2(INODE)*TREFFL
          PRESSD = PRESG2(INODE)*PRESFL

          BEE    = SYSHFS + (TEMPD-TREFCH)*SYSCPS - PRESSD/RHOD
     1                    + 0.5*(TEMPD*TEMPD-TREFCH*TREFCH)*BIGAM
          BEE    = BEE/FMREFL + 0.5*VEL02U

          BEE    = BEE*RHORPR
          IF (INODE .LT. 30) write(6,*) ' bee new and old', bee,
     1           DPENG2(4,INODE)

          DPENG2(4,INODE) = BEE

30      CONTINUE

        RETURN
        END
```

CHKSP2

```
        SUBROUTINE CHKSP2 (LCELL,MEM1,MEM2,MEM3,MEM4,NERR,NAME)
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'HEXCOD.INC'
        INCLUDE 'IOCOMN.INC'
        CHARACTER NAME*6
        DIMENSION NLEV(0:MLVLG2), LEVEL(0:MLVLG2,MCELG2)
C
C***********************************************************************
C
C       THIS SUBROUTINE CHECKS THE ASSIGNMENTS OF THE SUPER-CELL
C       ARRAY.  THESE ASSIGNMENTS ARE PRONE TO ERROR AFTER THE GRID-
C       DIVIDE AND GRID-COLLAPSE ROUTINES.  HENCE THIS ROUTINE MUST BE
C       USED AS A DEBUG CHECK AFTER CALLS TO THESE GRID CHANGING ROUTINES
C       IS MADE. LCELL IS THE MOST RECENTLY DIVIDED OR COLLAPSED CELL;
C       WHERE THE ERROR MIGHT OCCUR.  MEM1 THRU MEM4 ARE THE SUBCELLS OF
C       LCELL IF IT WERE COLLAPSED.  NAME INDICATES WHEN AND WHERE
C       THE ERROR OCCURED.  NERR COUNTS NUMBER OF ERRORS.
C
C***********************************************************************
C
C       COUNT THE PREVIOUS NUMBER OF ERRORS AND SET DEBUG UNIT
C
```

```
      NERRP  = NERR
      OPEN (UNIT=JDUMY2, FILE='CHKSP2.DAT', STATUS='NEW')
C
C     INITIALIZE THE NUMBER OF CELLS IN EACH SPATIAL LEVEL
C
      DO 10 N = 0, NLVLG2
         NLEV(N) = 0
10    CONTINUE
C
C     FOR EACH CELL FIND THE LEVEL AND SAVE THE CELLS AT THE SAME
C     LEVEL TOGETHER.  NLEV(L) CONTAINS THE TOTAL NUMBER OF CELLS
C     AT LEVEL L, WHEREAS LEVEL(L,JCELL) CONATINS THE VALUE OF THE
C     JTH CELL AT LEVEL L.
C
      DO 20 ICELL = 1, NCELG2
         KX                = KAUXG2(ICELL)
         K5LEVG            = IAND(KX,KU000F)
         LEVELG            = ISHFT(K5LEVG,-16)
         NLEV(LEVELG)      = NLEV(LEVELG) + 1
         NUM               = NLEV(LEVELG)
         LEVEL(LEVELG,NUM) = ICELL
20    CONTINUE
C
C     NOW LOOP THROUGH ALL THE CELLS; IF FOR A CELL SUPERCELL DOES
C     NOT EXIST THEN IT IS EITHER A BASE CELL OR A FINE CELL (CEWIC)
C     IN WHICH CASE THERE IS NO NEED TO FIND ITS SUUPERCELL.  ONCE
C     A CELL IS IDENTIFIED AT A CERTAIN LEVEL THEN THE NODES OF THE
C     ALL THE CELLS AT A LOWER LEVEL ARE CHECKED.  IF THE IDENTIFIED
C     CELL (ICELL) AND A LOWER LEVEL CELL (JCELL) AGREE IN NODE
C     ASSIGNMENT THEN JCELL IS THE SUPERCELL OF ICELL.  ISUPAS
C     IS THE ASSIGNED SUPERCELL; ISUPCL IS THE CALCULATED ONE.
C
      DO 40 ICELL = 1, NCELG2
         ISUPAS = ICELG2(10,ICELL)
         IF (ISUPAS .EQ. 0) GOTO 40
         KX     = KAUXG2(ICELL)
         K5LEVG = IAND(KX,KU000F)
         LEVELG = ISHFT(K5LEVG,-16)
         IF (LEVELG .EQ. 0) GOTO 40
         LEVELS = LEVELG-1
         ISUPCL = 0
         ISW    = ICELG2(2,ICELL)
         ISE    = ICELG2(4,ICELL)
         INE    = ICELG2(6,ICELL)
         INW    = ICELG2(8,ICELL)
         DO 30 KCELL = 1, NLEV(LEVELS)
            JCELL = LEVEL(LEVELS,KCELL)
            JSW   = ICELG2(2,JCELL)
            JSE   = ICELG2(4,JCELL)
            JNE   = ICELG2(6,JCELL)
            JNW   = ICELG2(8,JCELL)
            IF (ISW .EQ. JSW) THEN
               ISUPCL = JCELL
               GOTO 30
            ENDIF
            IF (ISE .EQ. JSE) THEN
               ISUPCL = JCELL
```

```
               GOTO 30
             ENDIF
           - IF (INE .EQ. JNE) THEN
               ISUPCL = JCELL
               GOTO 30
             ENDIF
             IF (INW .EQ. JNW) THEN
               ISUPCL = JCELL
               GOTO 30
             ENDIF
30         CONTINUE
C
           IF (ISUPCL .NE. ISUPAS) THEN
             NERR  = NERR + 1
             WRITE(JTERMO,1000) ICELL,ISW,ISE,INE,INW,ISUPAS,ISUPCL
             WRITE(JDUMY2,1000) ICELL,ISW,ISE,INE,INW,ISUPAS,ISUPCL
           ENDIF
C
C          GO BACK FOR NEXT CELL
C
40       CONTINUE
C
         IF (NERR .NE. NERRP) THEN
           WRITE(JTERMO,1100) NAME,NITRE2
           WRITE(JDUMY2,1100) NAME,NITRE2
           WRITE(JDUMY2,1200) LCELL, MEM1, MEM2, MEM3, MEM4
           CLOSE(UNIT=JDUMY2, DISP='KEEP')
         ELSE
           CLOSE(UNIT=JDUMY2, DISP='DELETE')
         ENDIF
C
         IF (NERR .NE. 0) THEN
           JPRINT = JDUMY3
           CALL G2PRNT(15)
           STOP 'ERROR IN CHECK ROUTINES; LOOK ALSO G2PRNT'
         ENDIF
C
C        -----------------
C        FORMAT STATEMENTS
C        -----------------
C
1000     FORMAT(2X,'ICELL =',I5,5X,'ISW =',I5,5X,'ISE =',I5,5X,'INE =',
     1         I5,5X,'INW =',I5,5X,'ISUPAS =',I5,5X,'ISUPCL =',I5)
1100     FORMAT(2X,'ERROR ',A6,5X,'AFTER',I5,2X,'ITERATIONS IN CHKSP2'/)
1200     FORMAT(2X,'LCELL =',I5,5X,'MEM1 =',I5,5X,'MEM2 =',I5,
     1                          5X,'MEM3 =',I5,5X,'MEM4 =',I5/)
C
         RETURN
         END
```

CHKTM2

```
         SUBROUTINE CHKTM2 (INODE)
```

```
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'

C*********************************************************************

C     THIS SUBROUTINE CORRECTS THE CONSERVATIVE VARIABLES AT A GIVEN
C     NODE 'INODE', IF THE PRESSURE OR TEMPERATURE AT THAT NODE BECOMES
C     NEGATIVE.  IT IS HOPED THAT SUCH A SITUATION ONLY OCCURS AT A FEW
C     NODES.  THE DEPENDENT VARIABLES ARE RESET TO THE VALUES OF THE
C     NEAREST NODE WHICH HAS POSITIVE PRESSURE AND TEMPERATURE.

C*********************************************************************

      WRITE(JDEBUG,*) ' NODE WITH NEG PR OR TEMP=',INODE
C
C     FIND THE SURROUNDING CELLS OF THIS NODE
C
      NB1 = NEIBG2(1,INODE)
      NB2 = NEIBG2(2,INODE)
      NB3 = NEIBG2(3,INODE)
      NB4 = NEIBG2(4,INODE)
      WRITE(JDEBUG,*) ' NB1 ETC',NB1,NB2,NB3,NB4
C
C     FIND THE SURROUNDING NODES OF THIS NODE
C
      NODES = 0
      NODEE = 0
      NODEN = 0
      NODEW = 0
C
      IF (NB1 .NE. 0) THEN
         NODES = ICELG2(4,NB1)
         NODEW = ICELG2(8,NB1)
      ENDIF
C
      IF (NB2 .NE. 0) THEN
         NODES = ICELG2(2,NB2)
         NODEE = ICELG2(6,NB2)
      ENDIF
C
      IF (NB3 .NE. 0) THEN
         NODEE = ICELG2(4,NB3)
         NODEN = ICELG2(8,NB3)
      ENDIF
C
      IF (NB4 .NE. 0) THEN
         NODEN = ICELG2(6,NB4)
         NODEW = ICELG2(2,NB4)
      ENDIF
      WRITE(JDEBUG,*) ' NODES ETC',NODES,NODEE,NODEN,NODEW
C
C     COMPUTE THE DISTANCES OF THESE NODES FROM THE CENTER NODE
C
```

```
      XS = 1.E10
      YS = 1.E10
      XE = 1.E10
      YE = 1.E10
      XN = 1.E10
      YN = 1.E10
      XW = 1.E10
      YW = 1.E10
C
      IF (NODES .NE. 0) THEN
         IF (PRESG2(NODES).GT.0. .AND. TEMPG2(NODES).GT.0.) THEN
            XS = GEOMG2(1,NODES)
            YS = GEOMG2(2,NODES)
         ENDIF
      ENDIF
C
      IF (NODEE .NE. 0) THEN
         IF (PRESG2(NODEE).GT.0. .AND. TEMPG2(NODEE).GT.0.) THEN
            XE = GEOMG2(1,NODEE)
            YE = GEOMG2(2,NODEE)
         ENDIF
      ENDIF
C
      IF (NODEN .NE. 0) THEN
         IF (PRESG2(NODEN).GT.0. .AND. TEMPG2(NODEN).GT.0.) THEN
            XN = GEOMG2(1,NODEN)
            YN = GEOMG2(2,NODEN)
         ENDIF
      ENDIF
C
      IF (NODEW .NE. 0) THEN
         IF (PRESG2(NODEW).GT.0. .AND. TEMPG2(NODEW).GT.0.) THEN
            XW = GEOMG2(1,NODEW)
            YW = GEOMG2(2,NODEW)
         ENDIF
      ENDIF
C
C     COMPUTE THE CURVILINEAR DISTANCE
C
      XS   = (GEOMG2(1,INODE)-XS)**2 + (GEOMG2(2,INODE)-YS)**2
      XE   = (GEOMG2(1,INODE)-XE)**2 + (GEOMG2(2,INODE)-YE)**2
      XN   = (GEOMG2(1,INODE)-XN)**2 + (GEOMG2(2,INODE)-YN)**2
      XW   = (GEOMG2(1,INODE)-XW)**2 + (GEOMG2(2,INODE)-YW)**2
      XMIN = 1.E8

      XMIN = MIN (XS, XE, XN, XW, XMIN)

      IF (XS .EQ. XMIN) THEN
         NODET = NODES
      ELSEIF (XE .EQ. XMIN) THEN
         NODET = NODEE
      ELSEIF (XN .EQ. XMIN) THEN
         NODET = NODEN
      ELSEIF (XW .EQ. XMIN) THEN
         NODET = NODEW
      ELSE
         WRITE (JDEBUG,*) ' ORPHAN NODE'
```

```
         RETURN
       ENDIF
C
C      NOW CORRECT THE NODE
C
       DO 10 IQ = 1, NEQNFL
          DPENG2(IQ,INODE) = DPENG2(IQ,NODET)
10     CONTINUE

       WRITE(JDEBUG,*) ' NEW NODE AND PRESSURE =',NODET,
     1                  PRESG2(NODET), PRESG2(INODE)
       WRITE(JDEBUG,*) ' NEW NODE AND TEMPERAT =',NODET,
     1                  TEMPG2(NODET), TEMPG2(INODE)

       PRESG2(INODE) = PRESG2(NODET)
       TEMPG2(INODE) = TEMPG2(NODET)

       RETURN
       END
```

CHKYMX

```
       SUBROUTINE CHKYMX

       INCLUDE 'PRECIS.INC'
       INCLUDE 'PARMV2.INC'
       INCLUDE 'CHCOMN.INC'
       INCLUDE 'G2COMN.INC'
       INCLUDE 'PRCOMN.INC'

C***********************************************************************

C      THIS SUBROUTINE COMPUTES THE MAXIMUM ALLOWABLE MASS-FRACTION FOR
C      EVERY SPECIES IN THE REACTION SYSTEM.  THIS ROUTINE IS SPECIALIZED
C      FOR ROGERS AND CHINITZ MODEL, WHEREAS FOR OTHER REACTION SYSTEM
C      SYSTEMS IT DOES A SIMPLER CALCULATION.  THE MAXIMUM ALLOWABLE
C      VALUES COMPUTED HERE CAN BE USED IN CONJUNCTION WITH THE ROUTINE
C      E2PRMT WHICH CHECKS IF A SPECIES IS TRYING TO CROSS THE ALLOWABLE
C      BOUNDS IN THE WHOLE SPATIO-TEMPORAL DOMAIN.

C***********************************************************************
C
C      FOR THE ROGER AND CHINITZ MODEL

       IF (KROGER .EQ. 1) THEN
          DO 10 IS = 1, NSPECH
             YMAXCH(IS) = 0.
10        CONTINUE
       ELSE
          DO 20 IS = 1, NEQSCH+1
             YMAXCH(IS) = 1. - YNRTCH
20        CONTINUE
          RETURN
```

```
      ENDIF
C
      YN2MIN = 1000.
      YN2MAX =-1000.
C
C     SCAN ALL THE BOUNDARY NODES FOR THE ROGERS AND CHINITZ MODEL
C
      DO 40 IBND = 1, NBNDG2
         INODE  = IBNDG2(1,IBND)
         IBCTYP = IBNDG2(5,IBND)
C
C        CHECK ONLY THE INFLOW BOUNDARY POINTS
C
         IF (IBCTYP .EQ. 2) THEN

            RHORPR = DPENG2(1,INODE)

C           COMPUTE THE MASS FRACTIONS FOR EACH SPECIES AND UPDATE
C           THE MAXIMUM IF NEED BE

            SUMY   = 0.
            YUPPER = 1. - YNRTCH

            DO 30 IS = 1, NEQSCH
              JS         = NEQBAS + IS
              YSPEPR(IS) = DPENG2(JS,INODE)/DPENG2(1,INODE)
              YMAXCH(IS) = MAX (YSPEPR(IS), YMAXCH(IS))
              SUMY       = SUMY + YSPEPR(IS)
30          CONTINUE

            YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
            IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
            YMAXCH(NEQSCH+1) = MAX(YSPEPR(NEQSCH+1),YMAXCH(NEQSCH+1))
C
C           MINIMUM AND MAXIMUM VALUES OF N2 WILL BE USED TO SEE IF
C           THE FLOW IS PRE-MIXED
            IF (NEQSCH .EQ. 4) THEN
              YN2MIN    = MIN(YSPEPR(5),YN2MIN)
              YN2MAX    = MAX(YSPEPR(5),YN2MAX)
            ENDIF

          ENDIF
40       CONTINUE

C       RATIO OF MOLE/MASS FOR OXYGEN AND HYDROGEN
C
      RMMO2 = YMAXCH(1)/AMWTCH(1)
      RMMH2 = YMAXCH(3)/AMWTCH(3)
      RMMOH = MIN (RMMO2, RMMH2)
      YMAXCH(2) = 2.*AMWTCH(2)*RMMOH

      RMMO2 = 2.*RMMO2
      RMMH20 = MIN (RMMO2, RMMH2)
      YMAXCH(4) = AMWTCH(4)*RMMH20

      IF (NEQSCH .EQ. 3) THEN
          YMAXCH(5)   = YNRTCH
```

```
            IALOCH(6,3) = -9
        ELSE
            IALOCH(6,3) = 0
            YMAXCH(5)   = YN2MAX
            IF (YN2MIN .EQ. YN2MAX) IALOCH(6,3) = -9
        ENDIF

        RETURN
        END
```

CPSPFU

```
        FUNCTION CPSP(ITYPE,T)

C***********************************************************************

C       THIS FUNCTION COMPUTES THE MOLAL SPECIFIC HEATS OF VARIOUS
C       SPECIES AS A FUNCTION OF TEMPERATURE IN DEGREE KELVIN.  THE
C       INPUT VARIABLE T IS TEMPERATURE/100.

C***********************************************************************

C       SPNAME(1 ) = 'N2      '
C       SPNAME(2 ) = 'O2      '
C       SPNAME(3 ) = 'H2      '
C       SPNAME(4 ) = 'CO      '
C       SPNAME(5 ) = 'OH      '
C       SPNAME(6 ) = 'NO      '
C       SPNAME(7 ) = 'H2O     '
C       SPNAME(8 ) = 'CO2     '
C       SPNAME(9 ) = 'NO2     '
C       SPNAME(10) = 'CH4     '
C       SPNAME(11) = 'C2H4    '
C       SPNAME(12) = 'C2H6    '
C       SPNAME(13) = 'C3H8    '
C       SPNAME(14) = 'C4H10   '

C
C       MOLAL SPECIFIC HEAT FOR N2 IN KJ/KMOL/K
C
        IF (ITYPE .GT. 1) GOTO 10
        CPSP=39.060 - 512.79/T**1.5 + 1072.7/T**2 - 820.4/T**3
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR O2 IN KJ/KMOL/K
C
10      IF (ITYPE .GT. 2) GOTO 20
        CPSP=37.432 + 0.020102*T**1.5 - 178.57/T**1.5 + 236.88/(T*T)
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR H2 IN KJ/KMOL/K
```

```
C
20      IF (ITYPE .GT. 3) GOTO 30
        CPSP=56.505 - 702.74/T**0.75 + 1165./T - 560.7/T**1.5
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR CO IN KJ/KMOL/K
C
30      IF (ITYPE .GT. 4) GOTO 40
        CPSP=69.145 - 0.70463*T**0.75 - 200.77/T**0.5 + 176.76/T**0.75
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR OH IN KJ/KMOL/K
C
40      IF (ITYPE .GT. 5) GOTO 50
        CPSP=81.546 - 59.350*T**0.25 + 17.329*T**0.75 - 4.266*T
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR NO IN KJ/KMOL/K
C
50      IF (ITYPE .GT. 6) GOTO 60
        CPSP=59.283 - 1.7096*T**0.5 - 70.613/T**0.5 + 74.889/T**1.5
        RETURN
C
C       MOLAL SPECIFIC HEAT FOR H2O IN KJ/KMOL/K
C
60      IF (ITYPE .GT. 7) GOTO 70
        CPSP=143.05 - 183.54*T**0.25 + 82.751*T**0.5 - 3.6989*T
        RETURN
C
C       MOLAL SPECIFIC HEAT FOR CO2 IN KJ/KMOL/K
C
70      IF (ITYPE .GT. 8) GOTO 80
        CPSP=-3.7357 + 30.529*T**0.5 - 4.1034*T + 0.024198*T**2
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR NO2 IN KJ/KMOL/K
C
80      IF (ITYPE .GT. 9) GOTO 90
        CPSP=46.045 + 216.1/T**0.5 - 363.66/T**0.75 + 232.55/T**2
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR CH4 IN KJ/KMOL/K
C
90      IF (ITYPE .GT. 10) GOTO 100
        CPSP=-672.87 + 439.74*T**0.25 - 24.875*T**0.75 + 323.88/T**0.5
        RETURN

C
C       MOLAL SPECIFIC HEAT FOR C2H4 IN KJ/KMOL/K
C
100     IF (ITYPE .GT. 11) GOTO 110
        CPSP=-95.395 + 123.15*T**0.5 - 35.641*T**0.75 + 182.77/T**3
```

```
      RETURN
C
C     MOLAL SPECIFIC HEAT FOR C2H6 IN KJ/KMOL/K
C
110   IF (ITYPE .GT. 12) GOTO 120
      CPSP=6.895 + 17.26*T - 0.6402*T**2 + 0.00728*T**3
      RETURN

C
C     MOLAL SPECIFIC HEAT FOR C3H8 IN KJ/KMOL/K
C
120   IF (ITYPE .GT. 13) GOTO 130
      CPSP=-4.042 + 30.46*T - 1.571*T**2 + 0.03171*T**3
      RETURN

C
C     MOLAL SPECIFIC HEAT FOR C4H10 IN KJ/KMOL/K
C
130   IF (ITYPE .GT. 14) GOTO 140
      CPSP=3.954 + 37.12*T - 1.833*T**2 + 0.03498*T**3
      RETURN

C     ADD OTHER OPTIONS HERE LATTER

140   RETURN

      END
```

DPINI2

```
      SUBROUTINE DPINI2
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
      INCLUDE 'PRCOMN.INC'
C
C**********************************************************************

C     THIS SUBROUTINE INITIALIZES THE DEPENDENT VARIABLES OVER ALL THE
C     NODES TO A UNIFORM FLOW, AND OTHER OPTIONS.

C**********************************************************************

      KDPENI = IPASKY(19)
C
C     ERROR CHECK; TYPE OF I.C. SELECTOR
C
```

```
      IF (KDPENI .LT. 0 .OR. KDPENI .GT. 3) THEN
         ZER1 = KDPENI
         ZER2 = 3.
         CALL ERRORM (42,'ERINIT','KDPENI',ZER1,'MAXVAL',ZER2,JPRINT,
     1     'ERROR IN INITIAL CONDITION SELECTOR')
      ENDIF

C     SET THE FINAL MASS FRACTIONS YSPEPR(S) FOR ALL SPECIES S
C     INCLUDING INERT ONES, ALSO INITIALIZE THE RECIPROCALS OF
C     MOLECULAR MASS FOR EACH SPECIES

      DO 10 IS = 1, NSPECH
          YSPEPR(IS) = YSPECH(IS)
          RAMWCH(IS) = 1./AMWTCH(IS)
10    CONTINUE
C
C     SEE IF YOU WANT TO READ ALL THE DEPENDENT VARIABLES FROM THE
C     FILE  INPUTD.DAT
C
      IF (KDPENI .EQ. 1) THEN
        DO 20 IN = 1, NNODG2
           READ (JREADD,1000) (DPENG2(K,IN), K = 1, NEQNFL)
20      CONTINUE
        CLOSE (JREADD)
        GOTO 130
      ENDIF
C
C     NOW SEE IF YOU WANT TO SET A UNIFORM DEPENDENT VARIABLES SET
C     OR A LINEARLY VARYING ONE
C
C     THE INDEPENDENT NORMALIZING QUANTITIES ARE
C     RHOI, PRESSURE, MACH #  AND SPECIES FRACTIONS
C
      RHOI   = RHORFR
      PRESSI = PRESFR
      RHOE   = RHORFR
      PRESSE = PRESFR
      AMACHE = AMCHFL
      UCOMPI = UCOMFR
      UCOMPE = UCOMFR
      VCOMPI = VCOMFR
      VCOMPE = VCOMFR
      BEI    = DPENFR(4)/RHOI
      BEE    = BEI
      VELO2I = UCOMPI*UCOMPI + VCOMPI*VCOMPI
      VELO2E = VELO2I
C
C     READ THE (DIMENSIONAL) CONDITIONS AT EXIT IF NECESSARY FOR
C     ALL SPECIES.  MAKE SURE THAT THE INERT SPECIES ARE INPUTTD
C     THE SAME AS IN C2INIT.  NOTE THAT THE FINAL MACH NUMBER IS
C     BASED UPON U-COMPONENT ONLY
C
      IF (KDPENI .GE. 3) THEN
C       READ THE FOLLOWING DIEMNSIONLESS QUANTITIES
        READ (JREADF,*) RHOE, PRESSE, AMACHE, VCOMPE
        DO 30 ISP = 1, NSPECH
           READ(JREADF,*) IS, YSPEPR(IS)
```

```
30        CONTINUE
          CLOSE (JREADF)
        ELSE
          GO TO 50
        ENDIF
C
C       COMPUTE THE INLET AND EXIT DEPENDENT VARIABLES
C
        SYSHFE = 0.
        SYSCPE = 0.
        SYSBMS = 0.
        BIGAM  = 0.

C       COMPUTE THE ENERGY TERM, COMPONENTS OF VELOCITY ETC.

        DO 40 IS = 1, NSPECH
            SYSHFE = SYSHFE + YSPEPR(IS)*FMHTCH(IS)
            SYSCPE = SYSCPE + YSPEPR(IS)*SPCPCH(IS)
            SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
            BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
40      CONTINUE
C
        UGASCO = UGASFL*SYSBMS
        TEMPE  = (PRESSE*PRESFL)/(UGASCO*RHOE*RHORFL)
        BIGAMT = BIGAM*TEMPE
        SYSCVE = SYSCPE + BIGAMT - UGASFL*SYSBMS
        GAMMAE = (SYSCPE + BIGAMT)/SYSCVE
        SOUNDE = GAMMAE*PRESSE/RHOE
        UCOMPE = AMACHE*SQRT(SOUNDE)
        VEL02E = UCOMPE*UCOMPE + VCOMPE*VCOMPE
        BEE    = SYSHFE + (TEMPE-TREFCH)*SYSCPE - UGASFL*TEMPE*SYSBMS
     1                  + 0.5*(TEMPE*TEMPE-TREFCH*TREFCH)*BIGAM
        BEE    = BEE/FMREFL + 0.5*VEL02E

50      U1I    = RHOI
        U1E    = RHOE
        U2I    = U1I*UCOMPI
        U2E    = U1E*UCOMPE
        U3I    = U1I*VCOMPI
        U3E    = U1E*VCOMPE
        U4I    = BEI*RHOI
        U4E    = BEE*RHOE
C
        DO 60 IS = 1, NEQSCH
          YSPEPR(IS) = YSPEPR(IS)*U1E - YSPECH(IS)*U1I
60      CONTINUE
C
        DU1 = U1E - U1I
        DU2 = U2E - U2I
        DU3 = U3E - U3I
        DU4 = U4E - U4I

C       IF KSRTE2 EQUALS 0 THEN GRID INITIALIZATION WAS DONE AN
C       ALGEBRAIC GRID GENERATOR.
C       LOCATE INLET AND EXIT OF THE RECTANGULAR DOMAIN BY READING
C       FROM INPUTG.DAT;  WE ASSUME THAT THERE ARE NO EMBEDDED CELLS
C       IN THE DOMAIN FOR THIS INITIALIZATION
```

```
C                                                          NX=NXRECT
C                                                          NY=NYRECT
C          -           L  L  L
C                      +  +  + . . .
C                      1  2  3
C           1+(NY-1)*NX +--+--+--+--+--+--+--+ NY*NX
C                      +       NORTH          + (NY-1)*NX = L
C                      + W                 E  +
C                      + E                 A  +
C                      + S                 S  + ...
C                 ...  + S                 S  +
C               1+2*NX + T                 T  + 3*NX
C                 1+NX +       SOUTH         + 2*NX
C                    1 +--+--+--+--+--+--+--+ NX
C                         2  3  ...     NX-1
C
C
      IF (KSRTE2 .EQ. 0 .OR. KSRTE2 .EQ. 1000) THEN
         REWIND (JREADG)
         READ (JREADG,1100) NXRECT, NYRECT
         CLOSE (JREADG)

C        SET ALL THE DEPENDENT VARIABLES

         DO 90 J = 1, NYRECT
            IBEG = 1 + (J-1)*NXRECT
            IEND = J*NXRECT
            XMIN = GEOMG2(1,IBEG)
            XMAX = GEOMG2(1,IEND)
            DX   = XMAX - XMIN
            DO 80 I = IBEG, IEND
              XDIS = GEOMG2(1,I)
              ALAM = (XDIS-XMIN)/DX

              IF (KDPENI .EQ. 2) ALAM = 0.
              DPENG2(1,I) = U1I + ALAM*DU1
              DPENG2(2,I) = U2I + ALAM*DU2
              DPENG2(3,I) = U3I + ALAM*DU3
              DPENG2(4,I) = U4I + ALAM*DU4

              DO 70 JS = NEQBAS+1, NEQNFL
                IS            = JS - NEQBAS
                DPENG2(JS,I) = YSPECH(IS)*U1I + ALAM*YSPEPR(IS)
70            CONTINUE
80          CONTINUE
90       CONTINUE
      ENDIF
C
C     IF KSRTE2 EQUALS 3 THEN GRID INITIALIZATION WAS DONE AN
C     THE ALGEBRAIC BLOCK GRID GENERATOR.
C     COMPUTE MINIMUM AND MAXIMUM X-DISTANCE FOR THE WHOLE DOMAIN
C     FOR THE LINEAR TYPE OF INTERPOLATION;  WE ASSUME THAT THERE
C     ARE NO EMBEDDED CELLS IN THE DOMAIN FOR THIS INITIALIZATION
C
      IF (KSRTE2 .EQ. 3 .OR. KSRTE2 .EQ. 1003) THEN

C        SET ALL THE DEPENDENT VARIABLES
```

```
            XMIN = 1.E6
            XMAX =-1.E6

            DO 100 IN = 1, NNODG2
               XMIN = MIN (XMIN,GEOMG2(1,IN))
               XMAX = MAX (XMAX,GEOMG2(1,IN))
100         CONTINUE

            DX   = XMAX - XMIN
            DO 120 I = 1, NNODG2
               XDIS = GEOMG2(1,I)
               ALAM = (XDIS-XMIN)/DX
               IF (KDPENI .EQ. 2) ALAM = 0.
               DPENG2(1,I) = U1I + ALAM*DU1
               DPENG2(2,I) = U2I + ALAM*DU2
               DPENG2(3,I) = U3I + ALAM*DU3
               DPENG2(4,I) = U4I + ALAM*DU4
               DO 110 JS = NEQBAS+1, NEQNFL
                 IS            = JS - NEQBAS
                 DPENG2(JS,I) = YSPECH(IS)*U1I + ALAM*YSPEPR(IS)
110            CONTINUE
120         CONTINUE

         ENDIF

130      CONTINUE

C        COMPUTE THE PRESSURE AT ALL THE NODES

         DO 140 IN = 1, NNODG2
C           CALL E2FLUX(IN)
            CALL E2PRMT(IN,1)
140      CONTINUE

C        COMPUTE ALL THE JACOBIAN TERMS AT ALL THE NODES

C        DO 150 IN = 1, NNODG2
C           CALL E2JACO(IN)
C150     CONTINUE

C        RESET THE FINAL FRACTIONS YSPEPR(S) FOR ALL SPECIES S, BECAUSE
C        THE JACOBIAN ROUTINES WILL CHANGE IT

         DO 160 IS = 1, NSPECH
             YSPEPR(IS) = YSPECH(IS)
160      CONTINUE

C        SAVE THE POINTER SYSTEM FOR THE INITIAL CONDITION ?

         IF (KDPENI .NE. 1) THEN
           WRITE(JTERMO,1200)
           READ(JTERMI,*) ITYPE
           IF (ITYPE .EQ. 1) THEN
             IF (KSRTE2 .LT. 1000) THEN
               CALL PSWRT2 (JPNTWR)
             ELSE
               CALL PSWRTU (JPNTWR)
```

```
            ENDIF
            STOP' POINTER SYSTEM DETAILS WRITTEN ON JPNTRE.DAT'
          ENDIF
        ENDIF
C
C       DEBUG PRINT
C
        IF (IDBGFL .NE. 5 .AND. IDBGFL .LT. 1000)  RETURN

        WRITE(JDEBUG,1300)
        WRITE(JDEBUG,1400)
        WRITE(JDEBUG,1500)
        WRITE(JDEBUG,1600) RHOI, UCOMPI, VCOMPI, PRESSI, BEI, AMCHFL
        WRITE(JDEBUG,1700)

        DO 170 IS = 1, NSPECH
          WRITE(JDEBUG,1800) IS, YSPECH(IS), YSPEPR(IS)
170     CONTINUE

        WRITE(JDEBUG,1900) RHOE, UCOMPE, VCOMPE, PRESSE, BEE, AMACHE
        WRITE(JDEBUG,2000)

        DO 180 IN = 1, NNODG2
          WRITE(JDEBUG,2100) IN, (DPENG2(K,IN), K = 1, NEQNFL)
180     CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(8G15.7)
1100    FORMAT(2I5)
1200    FORMAT(5X,'INPUT ONE OF THE FOLLOWING :'/
     1         10X,'1. SAVE THE INITIAL CONDITION POINTER SYSTEM'/
     2         10X,'2. RUN FURTHER WITHOUT SAVING POINTER SYSTEM'/
     3         10X,'===> ')
1300    FORMAT(//10X,'-----------------------' )
1400    FORMAT(  10X,'DEBUG PRINT FROM DPINI2' )
1500    FORMAT(  10X,'-----------------------'/)
1600    FORMAT(5X,'RHOI   = ',G14.5, 10X,'UCOMPI = ',G14.5/
     1          5X,'VCOMPI = ',G14.5, 10X,'PRESSI = ',G14.5/
     2          5X,'BEI    = ',G14.5, 10X,'MACHFL = ',G14.5/)
1700    FORMAT(/5X,'SPECIES',3X,'MASS FRACT_CH',3X,'MASS FRACT_PR')
1800    FORMAT(  5X, I5, 2X, G14.5, 5X, G14.5)
1900    FORMAT(5X,'RHOE   = ',G14.5, 10X,'UCOMPE = ',G14.5/
     1          5X,'VCOMPE = ',G14.5, 10X,'PRESSE = ',G14.5/
     2          5X,'BEE    = ',G14.5, 10X,'MACHE  = ',G14.5/)
2000    FORMAT(/5X, 'DEPENDENT VARIABLES'/7X, 'NODE',
     1           2X, 'DPEN1',6X,'DPEN2',7X,'DPEN3',7X,'DPEN4')
2100    FORMAT(5X,I5,8G12.5)

        RETURN
        END
```

E2BCNF

```
      SUBROUTINE E2BCNO (ITGL)
C                E2BCNF

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'H2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'JACOMN.INC'
      INCLUDE 'PRCOMN.INC'
      INCLUDE 'TICOMN.INC'
      DIMENSION EIGENU(MEQNFL), EIGENW(MEQNFL), ALVECT(MEQNFL,MEQNFL),
     1          DPENSV(MEQNFL), BIGWSV(MEQNFL)

C*********************************************************************

C     THIS SUBROUTINE APPLIES THE BOUNDARY CONDITIONS TO THE BOUNDARY
C     NODES ( NOT CONCERNED WITH CELLS)
C     THE TYPE OF BOUNDARY CONDITIONS ARE :-
C        1: RADIATION : SUPERSONIC EXIT
C        2: DIRECHLET : SUPERSONIC INLET
C        3: SOLID WALL BOUNDARY
C        4: INFLOW/OUTFLOW DETERMINATION
C        5: SUBSONIC INFLOW
C        6: SUBSONIC OUTFLOW

C*********************************************************************
C
      trgtmp = trigch/treffl
C     APPLY BOUNDARY CONDITIONS AT EACH BOUNDARY NODE

      DO 1600 IBOUND = 1, NBNDG2

C       BRANCH OUT ACCORDING TO TYPE
C       INODE IS THE BOUNDARY NODE
C       IONE  IS THE FIRST  CELL ADJACENT TO THE BOUNDARY NODE
C       ITWO  IS THE SECOND CELL ADJACENT TO THE BOUNDARY NODE
C       IEDGE IS 2 FOR SW CORNER, 4 FOR SE CORNER, ETC
C       ITYPE IS THE BOUNDARY CONDITION TYPE (1 THROUGH 6)

        INODE  = IBNDG2(1,IBOUND)
        IONE   = IBNDG2(2,IBOUND)
        ITWO   = IBNDG2(3,IBOUND)
        IEDGE  = IBNDG2(4,IBOUND)
        ITYPE  = IBNDG2(5,IBOUND)
C
C       SKIP TO NEXT BOUNDARY NODE IF THE TWO ADJACENT CELLS ARE
C       NOT AT THE CORRECT TEMPORAL LEVEL
C
C       GO TO (100,200,300,400,500,600,600,800,900,1000,1100),ITYPE
```

```
      GO TO (1600,200,300,400,500,600,600,800,900,1000,1100),ITYPE
      GO TO 1600
C
C     ------------------
C     RADIATION CONDITION
C     ------------------
C
C     SUPERSONIC EXIT -- DO NOTHING

100   GO TO 1600
C
C     ------------------
C     DIRECHLET CONDITION
C     ------------------
C
C     HOLD ALL CONDITIONS  --  SUPERSONIC INLET

200   DO 210 IQ = 1, NEQNFL
        CHNGE2(IQ,INODE) = 0.
210   CONTINUE
      if (tempg2(inode) .lt. trgtmp .and. kroger .eq. 1) then
          dpeng2(6,inode) = 0.
          dpeng2(8,inode) = 0.
      endif
      GO TO 1600
C
C     ------------------
C     SOLID WALL BOUNDARY
C     ------------------
C
300   IFACTR = 2

      GO TO (305,310,315,320,325,330,335,340), (IEDGE-1)
      GO TO 1600

C2    SOUTHWESTERN CORNER

305   KNODE1 = ICELG2(8,IONE)
      KNODE2 = ICELG2(4,IONE)
      INBND1 = NBCPG2(1,1)
      INBND2 = NBCPG2(1,2)
      IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
      IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
      IFACTR = 4
      GO TO 345

C3    SOUTHERN EDGE

310   KNODE1 = ICELG2(2,IONE)
      KNODE2 = ICELG2(4,ITWO)

      GO TO 345

C4    SOUTHEASTERN CORNER

315   KNODE1 = ICELG2(2,IONE)
      KNODE2 = ICELG2(6,IONE)
```

```
      INBND1 = NBCPG2(2,1)
      INBND2 = NBCPG2(2,2)
      IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
      IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
      IFACTR = 4
      GO TO 345

C5    EASTERN EDGE

320   KNODE1 = ICELG2(4,IONE)
      KNODE2 = ICELG2(6,ITWO)
      GO TO 345

C6    NORTHEASTERN CORNER

325   KNODE1 = ICELG2(4,IONE)
      KNODE2 = ICELG2(8,IONE)
      INBND1 = NBCPG2(3,1)
      INBND2 = NBCPG2(3,2)
      IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
      IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
      IFACTR = 4
      GO TO 345

C7    NORTHERN EDGE

330   KNODE1 = ICELG2(6,IONE)
      KNODE2 = ICELG2(8,ITWO)
      GO TO 345

C8    NORTHWESTERN CORNER

335   KNODE1 = ICELG2(6,IONE)
      KNODE2 = ICELG2(2,IONE)
      INBND1 = NBCPG2(4,1)
      INBND2 = NBCPG2(4,2)
      IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
      IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
      IFACTR = 4
      GO TO 345

C9    WESTERN EDGE

340   KNODE1 = ICELG2(8,IONE)
      KNODE2 = ICELG2(2,ITWO)

C     DETERMINE THE ANGLE OF THE SURFACE

345   DXSIDE = GEOMG2(1,KNODE2) - GEOMG2(1,KNODE1)
      DYSIDE = GEOMG2(2,KNODE2) - GEOMG2(2,KNODE1)
      PHYPO  = SQRT(DXSIDE*DXSIDE + DYSIDE*DYSIDE)
      COSANG = DXSIDE/PHYPO
      SINANG = DYSIDE/PHYPO

C     CALL E2BCSW (IONE,ITWO,INODE,IBOUND)

      DO 350 IQ = 1, NEQNFL
```

```
          CHNGE2(IQ,INODE) = IFACTR*CHNGE2(IQ,INODE)
350       CONTINUE

          RHO    = DPENG2(1,INODE) + CHNGE2(1,INODE)
          RHOU   = DPENG2(2,INODE) + CHNGE2(2,INODE)
          RHOV   = DPENG2(3,INODE) + CHNGE2(3,INODE)
          RHOQ   = RHOU*COSANG + RHOV*SINANG
          U2NEXT = RHOQ*COSANG
          U3NEXT = RHOQ*SINANG

          CHNGE2(2,INODE) = U2NEXT - DPENG2(2,INODE)
          CHNGE2(3,INODE) = U3NEXT - DPENG2(3,INODE)

          GO TO 1600
C
C         ----------------------------
C         INFLOW/OUTFLOW DETERMINATION
C         ----------------------------
C
C         DETERMINE IF THE FLOW IS ENTERING OR LEAVING THE
C         COMPUTATIONAL DOMAIN AND APPLY THE CHARACTERISTIC
C         BOUNDARY CONDITIONS ACCORDINGLY
C
400       DO 401 IQ = 1, NEQNFL
            DPENJA(IQ) = DPENG2(IQ,INODE) + CHNGE2(IQ,INODE)
401       CONTINUE
          GO TO (405,410,415,420,425,430,435,440), (IEDGE-1)
          GO TO 1600

C2        SOUTHWESTERN CORNER

405       KNODE1 = ICELG2(8,IONE)
          KNODE2 = ICELG2(4,IONE)
          INBND1 = NBCPG2(1,1)
          INBND2 = NBCPG2(1,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C3        SOUTHERN EDGE

410       KNODE1 = ICELG2(2,IONE)
          KNODE2 = ICELG2(4,ITWO)
          GO TO 445

C4        SOUTHEASTERN CORNER

415       KNODE1 = ICELG2(2,IONE)
          KNODE2 = ICELG2(6,IONE)
          INBND1 = NBCPG2(2,1)
          INBND2 = NBCPG2(2,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C5        EASTERN EDGE
```

```
420       KNODE1 = ICELG2(4,IONE)
          KNODE2 = ICELG2(6,ITWO)
          GO-TO 445

C6        NORTHEASTERN CORNER

425       KNODE1 = ICELG2(4,IONE)
          KNODE2 = ICELG2(8,IONE)
          INBND1 = NBCPG2(3,1)
          INBND2 = NBCPG2(3,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C7        NORTHERN EDGE

430       KNODE1 = ICELG2(6,IONE)
          KNODE2 = ICELG2(8,ITWO)
          GO TO 445

C8        NORTHWESTERN CORNER

435       KNODE1 = ICELG2(6,IONE)
          KNODE2 = ICELG2(2,IONE)
          INBND1 = NBCPG2(4,1)
          INBND2 = NBCPG2(4,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C9        WESTERN EDGE

440       KNODE1 = ICELG2(8,IONE)
          KNODE2 = ICELG2(2,ITWO)

C         DETERMINE ANGLE OF THE SURFACE; AND VELOCITY COMPONENTS

445       DXSIDE = GEOMG2(1,KNODE1) - GEOMG2(1,KNODE2)
          DYSIDE = GEOMG2(2,KNODE1) - GEOMG2(2,KNODE2)
          PHYPO  = SQRT(DXSIDE*DXSIDE + DYSIDE*DYSIDE)
          COSANG = DXSIDE/PHYPO
          SINANG = DYSIDE/PHYPO
          UCOMPR = DPENJA(2)
          VCOMPR = DPENJA(3)
C
C         COMPUTE THE NORMAL COMPONENT OF VELOCITY; IF POSITIVE
C         WE HAVE INFLOW; OTHERWISE OUTFLOW
C
          RHOQ   = UCOMPR*SINANG - VCOMPR*COSANG
C
          IF (RHOQ .GE. 0.) THEN
             ITYPE = 5
             GO TO 500
          ELSE
             ITYPE = 6
             GO TO 600
          ENDIF
```

```
C
      GO TO 1600
C
C     --------------------
C     CHARACTERISTIC INFLOW
C     --------------------
C     REVISE THE FOLLOWING ONCE IT STARTS WORKING
C
500   IF (ITWO .NE. 0) GOTO 504
      IF (IEDGE .EQ. 2) THEN
        DO 502 IQ = 1, NEQNFL
          DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(8,IONE))
          CHNGE2(IQ,INODE) = 0.
502     CONTINUE
        GOTO 1600
      ENDIF
C
      IF (IEDGE .EQ. 8) THEN
        DO 503 IQ = 1, NEQNFL
          DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(2,IONE))
          CHNGE2(IQ,INODE) = 0.
503     CONTINUE
        GOTO 1600
      ENDIF
C
C     SET UP THE DEPENDENT VARIABLES
C
504   DO 505 IQ = 1, NEQNFL
        DPENJA(IQ) = DPENG2(IQ,INODE)
505   CONTINUE
C
C     COMPUTE THE VELOCITY COMPONENTS, PRESSURE, GAMMA ETC
C
      CALL FLBGF2
C
      QVELO  = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
      SONDPR = SQRT(GAMAPR*PRESPR/RHORPR)
C
C     DETERMINE IF SUPERSONIC INLET
C
      IF (QVELO .GE. SONDPR) GO TO 200
C
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AT
C     THE BOUNDARY NODE AND COMPUTE EIGENVECTOR MATRIX IN THIS
C     ROTATED SYSTEM

      COSANG = UCOMPR/QVELO
      SINANG = VCOMPR/QVELO
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
      CALL E2VECT(ALVECT)

C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C     IS EMANATING

      PHYPO  = (SONDPR - QVELO)*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) + PHYPO*COSANG
```

```
      YPNT   = GEOMG2(2,INODE) + PHYPO*SINANG
C
C     FIND IN WHICH CELL THE POINT (XPNT,YPNT) IS LOCATED AND
C     INTERPOLATE IN CARTESIAN COORDINATES AT THIS POINT STORING
C     VALUES IN DPENJA(*)
C
      CALL G2LCAT (IBOUND, XPNT, YPNT)
C
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AGAIN
C
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
C
C     COMPUTE SOURCE TERMS AT THE INTERIOR POINT (ROTATED SYSTEM)
C
      CALL FRSOUR
C
C     COMPUTE THE PRODUCTS LU AND LW
C
      EIGENU(1) = 0.
      EIGENW(1) = 0.

      DO 510 JQ = 1, NEQNFL
        EIGENU(1) = EIGENU(1) + ALVECT(1,JQ)*DPENJA(JQ)
        EIGENW(1) = EIGENW(1) + ALVECT(1,JQ)*BIGWJA(JQ)
510   CONTINUE

      EIGENU(1) = EIGENU(1) + EIGENW(1)*CELLTI(IONE)
C
C     NOW COMPUTE THE CHARACTERISTICS FROM EXTERIOR DOMAIN
C     FIRST SET THE DEPENDENT VARIABLES
C
      DO 515 IQ = 1, NEQNFL
         DPENJA(IQ) = DPENFR(IQ)
515   CONTINUE
C
C     ROTATE THESE VALUES ALONG THE STREAMLINE AT THE SAME ANGLE
C
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
C
C     NOW COMPUTE THE MATRIX PRODUCTS
C
      DO 525 IQ = 2, NEQNFL
        EIGENU(IQ) = 0.
        DO 520 JQ = 1, NEQNFL
          EIGENU(IQ) = EIGENU(IQ) + ALVECT(IQ,JQ)*DPENJA(JQ)
520     CONTINUE
525   CONTINUE
C
C     NOW INVERT THIS TO COMPUTE THE REAL VALUES
C
      CALL GAUSS2 (ALVECT,EIGENU,DPENJA,NEQNFL,MEQNFL)
C
C     RECOMPUTE THE VELOCITY SO THAT IT CAN BE DISTRIBUTED
C
      UCOMPR    = DPENJA(2)/DPENJA(1)
```

```
          VCOMPR    = DPENJA(3)/DPENJA(1)
          QVELO     = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
          COSNEW    = UCOMPR/QVELO
          SINNEW    = VCOMPR/QVELO
          SINDUM    = SINANG*COSNEW+COSANG*SINNEW
          COSANG    = COSANG*COSNEW-SINANG*SINNEW
          SINANG    = SINDUM
          UCOMPR    = QVELO*COSANG
          VCOMPR    = QVELO*SINANG
          DPENJA(2) = UCOMPR*DPENJA(1)
          DPENJA(3) = VCOMPR*DPENJA(1)

          DO 530 JQ = 1, NEQNFL
            CHNGE2(JQ,INODE) = DPENJA(JQ) - DPENG2(JQ,INODE)
530       CONTINUE
C
          GO TO 1600
C
C         ----------------------
C         CHARACTERISTIC OUTFLOW
C         ----------------------
C
C         REVISE THE FOLLOWING ONCE IT STARTS WORKING
C
600       IF (ITWO .NE. 0) GOTO 604
          IF (IEDGE .EQ. 4) THEN
            DO 602 IQ = 1, NEQNFL
              DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(6,IONE))
              CHNGE2(IQ,INODE) = 0.
602         CONTINUE
            GOTO 1600
          ENDIF
C
          IF (IEDGE .EQ. 6) THEN
            DO 603 IQ = 1, NEQNFL
              DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(4,IONE))
              CHNGE2(IQ,INODE) = 0.
603         CONTINUE
            GOTO 1600
          ENDIF
C
C         SET UP THE DEPENDENT VARIABLES
C
604       IFAC = 1
          DO 605 IQ = 1, NEQNFL
             DPENJA(IQ) = DPENG2(IQ,INODE) + IFAC*CHNGE2(IQ,INODE)
             DPENSV(IQ) = DPENJA(IQ)
605       CONTINUE
C
          IF (IEDGE .EQ. 3 .AND. DPENJA(3) .GT. 0.) THEN
              DPENJA(3)       = 0.
              DPENG2(3,INODE) = 0.
              CHNGE2(3,INODE) = 0.
          ENDIF
C
C         COMPUTE THE VELOCITY COMPONENTS, PRESSURE, GAMMA ETC
C
```

```
      CALL FLBGF2
C
      QVELO  = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
      SONDPR = SQRT(GAMAPR*PRESPR/RHORPR)

C     DETERMINE IF SUPERSONIC EXIT
C
      IF (QVELO .GT. SONDPR) GO TO 700
C
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AT
C     THE BOUNDARY NODE AND COMPUTE EIGENVECTOR MATRIX IN THIS
C     ROTATED SYSTEM

      COSANG    = UCOMPR/QVELO
      SINANG    = VCOMPR/QVELO
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
C     DPENF2    = QVELO*DPENFR(1)
      CALL E2VECT(ALVECT)
C
C     DETERMINE THE SOURCE TERMS AT THE BOUNDARY NODE; SAVE THEM
C
      CALL FRSOUR
      DO 610 IQ = 1, NEQNFL
         BIGWSV(IQ) = BIGWJA(IQ)
610   CONTINUE
C
C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C     WITH SPEED (U+A) IS EMANATING

      PHYPO  = (SONDPR + QVELO)*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) - PHYPO*COSANG
      YPNT   = GEOMG2(2,INODE) - PHYPO*SINANG
C
C     FIND IN WHICH CELL THE POINT (XPNT,YPNT) IS LOCATED AND
C     INTERPOLATE IN CARTESIAN COORDINATES AT THIS POINT STORING
C     VALUES IN DPENJA(*)
C
      CALL G2LCAT (IBOUND, XPNT, YPNT)
C
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AGAIN
C
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
      DPENSV(2) = DPENSV(2)*COSANG + DPENSV(3)*SINANG
      DPENSV(3) = 0.
C
C     COMPUTE SOURCE TERMS AT THE INTERIOR POINT (ROTATED SYSTEM)
C
      CALL FRSOUR
C
C     COMPUTE THE PRODUCTS LU AND LW
C
      EIGENU(2) = 0.
      EIGENW(2) = 0.

      DO 615 JQ = 1, NEQNFL
```

```
          EIGENU(2) = EIGENU(2) + ALVECT(2,JQ)*DPENJA(JQ)
          EIGENW(2) = EIGENW(2) + ALVECT(2,JQ)*BIGWJA(JQ)
615       CONTINUE

C         DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C         WITH SPEEDS U ARE EMANATING

          PHYPON = QVELO*CELLTI(IONE)
          XPNT   = GEOMG2(1,INODE) - PHYPON*COSANG
          YPNT   = GEOMG2(2,INODE) - PHYPON*SINANG
C
C         INTERPOLATE VALUES
C
          XL        = PHYPON/PHYPO
          XLM1      = 1. - XL
C
          DO 620 IQ = 1, NEQNFL
            DPENJA(IQ) = DPENJA(IQ)*XL + DPENSV(IQ)*XLM1
            BIGWJA(IQ) = BIGWJA(IQ)*XL + BIGWSV(IQ)*XLM1
620       CONTINUE
C
C         NOW COMPUTE THE MATRIX PRODUCTS
C
          DO 630 IQ = 3, NEQNFL
            EIGENU(IQ) = 0.
            EIGENW(IQ) = 0.
            DO 625 JQ = 1, NEQNFL
              EIGENU(IQ) = EIGENU(IQ) + ALVECT(IQ,JQ)*DPENJA(JQ)
              EIGENW(IQ) = EIGENW(IQ) + ALVECT(IQ,JQ)*BIGWJA(JQ)
625         CONTINUE
630       CONTINUE
C
C         CORRECT THE INTERIOR CHARACTERISTICS FOR TIME
C
          DO 635 IQ = 2, NEQNFL
            EIGENU(IQ) = EIGENU(IQ) + EIGENW(IQ)*CELLTI(IONE)
635       CONTINUE
C
C         NOW COMPUTE THE CHARACTERISTICS FROM EXTERIOR DOMAIN
C         FIRST SET THE DEPENDENT VARIABLES
C
          DO 640 IQ = 1, NEQNFL
             DPENJA(IQ) = DPENFR(IQ)
640       CONTINUE
C
C         ROTATE THESE VALUES ALONG THE STREAMLINE AT THE SAME ANGLE
C
          DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
C         DPENJA(2) = DPENF2
          DPENJA(3) = 0.

          EIGENU(1) = 0.
          DO 645 IQ = 1, NEQNFL
             EIGENU(1) = EIGENU(1) + ALVECT(1,IQ)*DPENJA(IQ)
645       CONTINUE
C
C         NOW INVERT THIS TO COMPUTE THE REAL VALUES
```

```
C
      CALL GAUSS2 (ALVECT,EIGENU,DPENJA,NEQNFL,MEQNFL)
C
C     RECOMPUTE THE VELOCITY SO THAT IT CAN BE DISTRIBUTED
C
      UCOMPR    = DPENJA(2)/DPENJA(1)
      VCOMPR    = DPENJA(3)/DPENJA(1)
      QVELO     = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
      UCOMPR    = QVELO*COSANG
      VCOMPR    = QVELO*SINANG
      DPENJA(2) = UCOMPR*DPENJA(1)
      DPENJA(3) = VCOMPR*DPENJA(1)

      DO 650 JQ = 1, NEQNFL
        CHNGE2(JQ,INODE) = DPENJA(JQ) - DPENG2(JQ,INODE)
650   CONTINUE
C
      GO TO 1600
C
C     ---------------------------------
C     CHARACTERISTIC SUPERSONIC OUTFLOW
C     ---------------------------------
C
C     DETERMINE THE SOURCE TERMS AT THE BOUNDARY NODE; SAVE THEM
C
700   CALL FRSOUR
      DO 705 IQ = 1, NEQNFL
         BIGWSV(IQ) = BIGWJA(IQ)
705   CONTINUE
C
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AT
C     THE BOUNDARY NODE AND COMPUTE EIGENVECTOR MATRIX IN THIS
C     ROTATED SYSTEM
C
      COSANG = UCOMPR/QVELO
      SINANG = VCOMPR/QVELO
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
      CALL E2VECT(ALVECT)
C
C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C     (U+A) IS EMANATING

      PHYPO  = (SONDPR + QVELO)*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) - PHYPO*COSANG
      YPNT   = GEOMG2(2,INODE) - PHYPO*SINANG
C
C     FIND IN WHICH CELL THE POINT (XPNT,YPNT) IS LOCATED AND
C     INTERPOLATE IN CARTESIAN COORDINATES AT THIS POINT STORING
C     VALUES IN DPENJA(*)
C
      CALL G2LCAT (IBOUND, XPNT, YPNT)
C
C     THE DIRECTION OF THE STREAM LINE MIGHT HAVE CHANGED, SO
C     CORRECT IT
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AGAIN
C
```

```
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
      DPENSV(2) = DPENSV(2)*COSANG + DPENSV(3)*SINANG
      DPENSV(3) = 0.
C
C     COMPUTE SOURCE TERMS AT THE INTERIOR POINT (ROTATED SYSTEM)
C
      CALL FRSOUR
C
C     COMPUTE THE PRODUCTS LU AND LW
C
      EIGENU(2) = 0.
      EIGENW(2) = 0.

      DO 710 JQ = 1, NEQNFL
        EIGENU(2) = EIGENU(2) + ALVECT(2,JQ)*DPENJA(JQ)
        EIGENW(2) = EIGENW(2) + ALVECT(2,JQ)*BIGWJA(JQ)
710   CONTINUE
C
C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTICS
C     WITH SPEEDS U ARE EMANATING
C
      PHYPON = QVELO*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) - PHYPON*COSANG
      YPNT   = GEOMG2(2,INODE) - PHYPON*SINANG
C
C     INTERPOLATE VALUES
C
      XL    = PHYPON/PHYPO
      XLM1  = 1. - XL
      PHYPO = PHYPON
C
      DO 715 IQ = 1, NEQNFL
        DPENJA(IQ) = DPENJA(IQ)*XL + DPENSV(IQ)*XLM1
        BIGWJA(IQ) = BIGWJA(IQ)*XL + BIGWSV(IQ)*XLM1
715   CONTINUE
C
C     NOW COMPUTE THE MATRIX PRODUCTS
C
      DO 725 IQ = 3, NEQNFL
        EIGENU(IQ) = 0.
        EIGENW(IQ) = 0.
        DO 720 JQ = 1, NEQNFL
          EIGENU(IQ) = EIGENU(IQ) + ALVECT(IQ,JQ)*DPENJA(JQ)
          EIGENW(IQ) = EIGENW(IQ) + ALVECT(IQ,JQ)*BIGWJA(JQ)
720     CONTINUE
725   CONTINUE
C
C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C     WITH SPEED (U-A) IS EMANATING

      PHYPON = (QVELO - SONDPR)*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) - PHYPON*COSANG
      YPNT   = GEOMG2(2,INODE) - PHYPON*SINANG
C
C     INTERPOLATE VALUES
C
```

```
          XL        = PHYPON/PHYPO
          XLM1      = 1. - XL
C
          DO 730 IQ = 1, NEQNFL
            DPENJA(IQ) = DPENJA(IQ)*XL + DPENSV(IQ)*XLM1
            BIGWJA(IQ) = BIGWJA(IQ)*XL + BIGWSV(IQ)*XLM1
730       CONTINUE
C
          EIGENU(1) = 0.
          EIGENW(1) = 0.
          DO 735 IQ = 1, NEQNFL
             EIGENU(1) = EIGENU(1) + ALVECT(1,IQ)*DPENJA(IQ)
             EIGENW(1) = EIGENW(1) + ALVECT(1,IQ)*BIGWJA(IQ)
735       CONTINUE
C
C         CORRECT THE INTERIOR CHARACTERISTICS FOR TIME
C
          DO 740 IQ = 1, NEQNFL
            EIGENU(IQ) = EIGENU(IQ) + EIGENW(IQ)*CELLTI(IONE)
740       CONTINUE
C
C         NOW INVERT THIS TO COMPUTE THE REAL VALUES
C
          CALL GAUSS2 (ALVECT,EIGENU,DPENJA,NEQNFL,MEQNFL)
C
C         RECOMPUTE THE VELOCITY SO THAT IT CAN BE DISTRIBUTED
C
          UCOMPR    = DPENJA(2)/DPENJA(1)
          VCOMPR    = DPENJA(3)/DPENJA(1)
          QVELO     = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
          UCOMPR    = QVELO*COSANG
          VCOMPR    = QVELO*SINANG
          DPENJA(2) = UCOMPR*DPENJA(1)
          DPENJA(3) = VCOMPR*DPENJA(1)

          DO 745 JQ = 1, NEQNFL
            CHNGE2(JQ,INODE) = DPENJA(JQ) - DPENG2(JQ,INODE)
745       CONTINUE

          GOTO 1600
C
C         --------------------
C         EQUILIBRIUM BOUNDARY
C         --------------------
C
800       CONTINUE

          JFAC   = 2
          IF (ITWO .EQ. 0) JFAC = 2*JFAC

810       RHORPR = DPENG2(1,INODE)
          RECDEN = 1./RHORPR
          UCOMPR = DPENG2(2,INODE)*RECDEN
          VCOMPR = DPENG2(3,INODE)*RECDEN
          VELO2U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C         COMPUTE THE DIMENSIONAL QUANTITIES
          TEMPPR = TEMPG2(INODE)*TREFFL
```

```
C
C         SET THE FIRST FOUR CHANGES AS ZERO'S
C
          CHNGE2(1,INODE) = 0.
          CHNGE2(2,INODE) = 0.
          CHNGE2(3,INODE) = 0.
          CHNGE2(4,INODE) = 0.

C         COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
          SUMY   = 0.
          YUPPER = 1. - YNRTCH

          DO 820 JS = NEQBAS+1, NEQNFL
             IS         = JS - NEQBAS
             DPENG2(JS,INODE) = DPENG2(JS,INODE) +JFAC*CHNGE2(JS,INODE)
             CHNGE2(JS,INODE) = 0.
             YSPEPR(IS) = DPENG2(JS,INODE)*RECDEN
             IF (YSPEPR(IS) .LT. 0.) THEN
                YSPEPR(IS)       = 0.
                DPENG2(JS,INODE) = 0.
             ENDIF
C            IF (YSPEPR(IS) .GT. YUPPER) THEN
C               YSPEPR(IS)       = YUPPER
C               DPENG2(JS,INODE) = YUPPER*DPENG2(1,INODE)
C            ENDIF
             SUMY       = SUMY + YSPEPR(IS)
820       CONTINUE

          YSPEPR(NEQSCH+1) = YUPPER - SUMY
C         YSPEPR(NEQSCH+1) = ABS(1. - SUMY - YNRTCH)
          IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C
C         COMPUTE THE ENTHALPY OF THE MIXTURE
C
          SYSHFS = 0.
          SYSCPS = 0.
          SYSBMS = 0.
          BIGAM  = 0.

          DO 830 IS = 1, NSPECH
             SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
             SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
             SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
             BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
830       CONTINUE

          ENTHAL = SYSHFS + SYSCPS*(TEMPPR-TREFCH) +
     1             0.5*BIGAM*(TEMPPR**2-TREFCH**2)
          ENTHAL = ENTHAL/FMREFL + 0.5*VELO2U

C         COMPUTE THE DIMENSIONLESS PRESSURE
C
          PRESG2(INODE) = RHORPR*TEMPG2(INODE)*AMWTFL*SYSBMS
C
C         COMPUTE THE FOURTH COMPONENT OF STATE VECTOR
C
          DPENG2(4,INODE) = RHORPR*ENTHAL - PRESG2(INODE)
```

```
          GOTO 1600
C
C         -------------------------
C         CORNER BOUNDARY CONDITIONS
C         -------------------------
C
900       CONTINUE

          IF (IEDGE .EQ. 2) THEN
             INEXTN = 8
          ELSE IF (IEDGE .EQ. 4) THEN
             INEXTN = 6
          ELSE IF (IEDGE .EQ. 6) THEN
             INEXTN = 4
          ELSE IF (IEDGE .EQ. 8) THEN
             INEXTN = 2
          ELSE
             GOTO 1600
          ENDIF
          NODENB = ICELG2(INEXTN,IONE)
c
          DO 910 JS = 1, NEQNFL
             DPENG2(JS,INODE) = DPENG2(JS,NODENB)
             CHNGE2(JS,INODE) = 0.
910       CONTINUE
          GOTO 1600
C
C         ------------------------------
C         RADIATION CONDITION + FACTOR 2
C         ------------------------------
C
C         SUPERSONIC EXIT
C
1000      DO 1010 IQ = 1, NEQNFL
            CHNGE2(IQ,INODE) = 2.*CHNGE2(IQ,INODE)
1010      CONTINUE

          GO TO 1600
C
C         --------------------
C         VISCOUS WALL BOUNDARY
C         --------------------
C
C1100     IFACTR = 2
1100      FFACTR = 2.

          GO TO (1105,1110,1105,1110,1105,1110,1105,1110), (IEDGE-1)
          GO TO 1600

C1105     IFACTR = 4
1105      FFACTR = 4.
1110      IF (IEDGE .EQ. 0) FFACTR = 4./3.

C         SET THE VELOCITIES ZERO AND OTHER VALUES AS REFLECTION
          DO 1120 IQ = 1, NEQNFL
C           CHNGE2(IQ,INODE) = IFACTR*CHNGE2(IQ,INODE)
```

```
            CHNGE2(IQ,INODE) = FFACTR*CHNGE2(IQ,INODE)
1120        CONTINUE

            CHNGE2(2,INODE) = 0.
            CHNGE2(3,INODE) = 0.
            DPENG2(2,INODE) = 0.
            DPENG2(3,INODE) = 0.

C           GO TO 1600
C
C           GO BACK FOR NEXT NODE

1600      CONTINUE

          RETURN
          END
```

E2BCN0

```
          SUBROUTINE E2BCN0 (ITGL)

          INCLUDE '[.INC] PRECIS.INC/LIST'
          INCLUDE '[.INC] PARMV2.INC/LIST'
          INCLUDE '[.INC] CHCOMN.INC/LIST'
          INCLUDE '[.INC] E2COMN.INC/LIST'
          INCLUDE '[.INC] FLCOMN.INC/LIST'
          INCLUDE '[.INC] FRCOMN.INC/LIST'
          INCLUDE '[.INC] G2COMN.INC/LIST'
          INCLUDE '[.INC] H2COMN.INC/LIST'
          INCLUDE '[.INC] IOCOMN.INC/LIST'
          INCLUDE '[.INC] JACOMN.INC/LIST'
          INCLUDE '[.INC] PRCOMN.INC/LIST'
          INCLUDE '[.INC] TICOMN.INC/LIST'
          DIMENSION EIGENU(MEQNFL), EIGENW(MEQNFL), ALVECT(MEQNFL,MEQNFL),
        1           DPENSV(MEQNFL), BIGWSV(MEQNFL)
          LOGICAL   IWRITE

C*********************************************************************

C         THIS SUBROUTINE APPLIES THE BOUNDARY CONDITIONS TO THE BOUNDARY
C         NODES ( NOT CONCERNED WITH CELLS)
C         THE TYPE OF BOUNDARY CONDITIONS ARE :-
C            1: RADIATION : SUPERSONIC EXIT
C            2: DIRECHLET : SUPERSONIC INLET
C            3: SOLID WALL BOUNDARY
C            4: INFLOW/OUTFLOW DETERMINATION
C            5: SUBSONIC INFLOW
C            6: SUBSONIC OUTFLOW

C*********************************************************************
          IFAC = 1
C
C         WANT DEBUG PRINT ?
```

```
C
      IWRITE  = IDBGE2 .EQ. 5 .OR. IDBGE2 .GT. 1000

C     APPLY BOUNDARY CONDITIONS AT EACH BOUNDARY NODE

      DO 1600 IBOUND = 1, NBNDG2

C       BRANCH OUT ACCORDING TO TYPE
C       INODE IS THE BOUNDARY NODE
C       IONE  IS THE FIRST  CELL ADJACENT TO THE BOUNDARY NODE
C       ITWO  IS THE SECOND CELL ADJACENT TO THE BOUNDARY NODE
C       IEDGE IS 2 FOR SW CORNER, 4 FOR SE CORNER, ETC
C       ITYPE IS THE BOUNDARY CONDITION TYPE (1 THROUGH 6)

        INODE  = IBNDG2(1,IBOUND)
        IONE   = IBNDG2(2,IBOUND)
        ITWO   = IBNDG2(3,IBOUND)
        IEDGE  = IBNDG2(4,IBOUND)
        ITYPE  = IBNDG2(5,IBOUND)
C
C       SKIP TO NEXT BOUNDARY NODE IF THE TWO ADJACENT CELLS ARE
C       NOT AT THE CORRECT TEMPORAL LEVEL
C
        IF (CHNGE2(1,INODE) .EQ. 0.) GOTO 1600
10      GO TO (100,200,300,400,500,600,600,800,900,1000,1100),ITYPE
        GO TO 1500
C
C       -------------------
C       RADIATION CONDITION
C       -------------------
C
C       SUPERSONIC EXIT -- DO NOTHING

100     GO TO 1500
C
C       -------------------
C       DIRECHLET CONDITION
C       -------------------
C
C       HOLD ALL CONDITIONS  --  SUPERSONIC INLET

200     DO 210 IQ = 1, NEQNFL
          CHNGE2(IQ,INODE) = 0.
210     CONTINUE

        GO TO 1500
C
C       -------------------
C       SOLID WALL BOUNDARY
C       -------------------
C
300     IFACTR = 2

        GO TO (305,310,315,320,325,330,335,340), (IEDGE-1)
        GO TO 1500

C2      SOUTHWESTERN CORNER
```

```
305       KNODE1 = ICELG2(8,IONE)
          KNODE2 = ICELG2(4,IONE)
          INBND1 = NBCPG2(1,1)
          INBND2 = NBCPG2(1,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          IFACTR = 4
          GO TO 345

C3        SOUTHERN EDGE

310       KNODE1 = ICELG2(2,IONE)
          KNODE2 = ICELG2(4,ITWO)

          GO TO 345

C4        SOUTHEASTERN CORNER

315       KNODE1 = ICELG2(2,IONE)
          KNODE2 = ICELG2(6,IONE)
          INBND1 = NBCPG2(2,1)
          INBND2 = NBCPG2(2,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          IFACTR = 4
          GO TO 345

C5        EASTERN EDGE

320       KNODE1 = ICELG2(4,IONE)
          KNODE2 = ICELG2(6,ITWO)
          GO TO 345

C6        NORTHEASTERN CORNER

325       KNODE1 = ICELG2(4,IONE)
          KNODE2 = ICELG2(8,IONE)
          INBND1 = NBCPG2(3,1)
          INBND2 = NBCPG2(3,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          IFACTR = 4
          GO TO 345

C7        NORTHERN EDGE

330       KNODE1 = ICELG2(6,IONE)
          KNODE2 = ICELG2(8,ITWO)
          GO TO 345

C8        NORTHWESTERN CORNER

335       KNODE1 = ICELG2(6,IONE)
          KNODE2 = ICELG2(2,IONE)
          INBND1 = NBCPG2(4,1)
          INBND2 = NBCPG2(4,2)
```

```
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          IFACTR = 4
          GO TO 345

C9        WESTERN EDGE

340       KNODE1 = ICELG2(8,IONE)
          KNODE2 = ICELG2(2,ITWO)

C         DETERMINE THE ANGLE OF THE SURFACE

345       DXSIDE = GEOMG2(1,KNODE2) - GEOMG2(1,KNODE1)
          DYSIDE = GEOMG2(2,KNODE2) - GEOMG2(2,KNODE1)
          PHYPO  = SQRT(DXSIDE*DXSIDE + DYSIDE*DYSIDE)
          COSANG = DXSIDE/PHYPO
          SINANG = DYSIDE/PHYPO

C         CALL E2BCSW (IONE,ITWO,INODE,IBOUND)

          DO 350 IQ = 1, NEQNFL
            CHNGE2(IQ,INODE) = IFACTR*CHNGE2(IQ,INODE)
350       CONTINUE

          RHO    = DPENG2(1,INODE) + CHNGE2(1,INODE)
          RHOU   = DPENG2(2,INODE) + CHNGE2(2,INODE)
          RHOV   = DPENG2(3,INODE) + CHNGE2(3,INODE)
          RHOQ   = RHOU*COSANG + RHOV*SINANG
          U2NEXT = RHOQ*COSANG
          U3NEXT = RHOQ*SINANG

          CHNGE2(2,INODE) = U2NEXT - DPENG2(2,INODE)
          CHNGE2(3,INODE) = U3NEXT - DPENG2(3,INODE)

          GO TO 1500
C
C         ---------------------------
C         INFLOW/OUTFLOW DETERMINATION
C         ---------------------------
C
C         DETERMINE IF THE FLOW IS ENTERING OR LEAVING THE
C         COMPUTATIONAL DOMAIN AND APPLY THE CHARACTERISTIC
C         BOUNDARY CONDITIONS ACCORDINGLY
C
400       DO 401 IQ = 1, NEQNFL
            DPENJA(IQ) = DPENG2(IQ,INODE) + CHNGE2(IQ,INODE)
401       CONTINUE
          GO TO (405,410,415,420,425,430,435,440), (IEDGE-1)
          GO TO 1500

C2        SOUTHWESTERN CORNER

405       KNODE1 = ICELG2(8,IONE)
          KNODE2 = ICELG2(4,IONE)
          INBND1 = NBCPG2(1,1)
          INBND2 = NBCPG2(1,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
```

```
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C3        SOUTHERN EDGE

410       KNODE1 = ICELG2(2,IONE)
          KNODE2 = ICELG2(4,ITWO)
          GO TO 445

C4        SOUTHEASTERN CORNER

415       KNODE1 = ICELG2(2,IONE)
          KNODE2 = ICELG2(6,IONE)
          INBND1 = NBCPG2(2,1)
          INBND2 = NBCPG2(2,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C5        EASTERN EDGE

420       KNODE1 = ICELG2(4,IONE)
          KNODE2 = ICELG2(6,ITWO)
          GO TO 445

C6        NORTHEASTERN CORNER

425       KNODE1 = ICELG2(4,IONE)
          KNODE2 = ICELG2(8,IONE)
          INBND1 = NBCPG2(3,1)
          INBND2 = NBCPG2(3,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C7        NORTHERN EDGE

430       KNODE1 = ICELG2(6,IONE)
          KNODE2 = ICELG2(8,ITWO)
          GO TO 445

C8        NORTHWESTERN CORNER

435       KNODE1 = ICELG2(6,IONE)
          KNODE2 = ICELG2(2,IONE)
          INBND1 = NBCPG2(4,1)
          INBND2 = NBCPG2(4,2)
          IF (ITYPE .EQ. IBNDG2(5,INBND1)) KNODE2 = INODE
          IF (ITYPE .EQ. IBNDG2(5,INBND2)) KNODE1 = INODE
          GO TO 445

C9        WESTERN EDGE

440       KNODE1 = ICELG2(8,IONE)
          KNODE2 = ICELG2(2,ITWO)

C         DETERMINE ANGLE OF THE SURFACE; AND VELOCITY COMPONENTS
```

```
445       DXSIDE = GEOMG2(1,KNODE1) - GEOMG2(1,KNODE2)
          DYSIDE = GEOMG2(2,KNODE1) - GEOMG2(2,KNODE2)
          PHYPO  = SQRT(DXSIDE*DXSIDE + DYSIDE*DYSIDE)
          COSANG = DXSIDE/PHYPO
          SINANG = DYSIDE/PHYPO
          UCOMPR = DPENJA(2)
          VCOMPR = DPENJA(3)
C
C         COMPUTE THE NORMAL COMPONENT OF VELOCITY; IF POSITIVE
C         WE HAVE INFLOW; OTHERWISE OUTFLOW
C
          RHOQ   = UCOMPR*SINANG - VCOMPR*COSANG
C
          IF (RHOQ .GE. 0.) THEN
             ITYPE = 5
             GO TO 500
          ELSE
             ITYPE = 6
             GO TO 600
          ENDIF
C
          GO TO 1500
C
C         ---------------------
C         CHARACTERISTIC INFLOW
C         ---------------------
C         REVISE THE FOLLOWING ONCE IT STARTS WORKING
C
500       IF (ITWO .NE. 0) GOTO 504
          IF (IEDGE .EQ. 2) THEN
            DO 502 IQ = 1, NEQNFL
              DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(8,IONE))
              CHNGE2(IQ,INODE) = 0.
502         CONTINUE
            GOTO 1500
          ENDIF
C
          IF (IEDGE .EQ. 8) THEN
            DO 503 IQ = 1, NEQNFL
              DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(2,IONE))
              CHNGE2(IQ,INODE) = 0.
503         CONTINUE
            GOTO 1500
          ENDIF
C
C         SET UP THE DEPENDENT VARIABLES
C
504       DO 505 IQ = 1, NEQNFL
            DPENJA(IQ) = DPENG2(IQ,INODE)
505       CONTINUE
C
C         COMPUTE THE VELOCITY COMPONENTS, PRESSURE, GAMMA ETC
C
          CALL FLBGF2
C
          QVELO  = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
```

```
      SONDPR = SQRT(GAMAPR*PRESPR/RHORPR)
C
C     DETERMINE IF SUPERSONIC INLET
C
      IF (QVELO .GE. SONDPR) GO TO 200
C
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AT
C     THE BOUNDARY NODE AND COMPUTE EIGENVECTOR MATRIX IN THIS
C     ROTATED SYSTEM

      COSANG = UCOMPR/QVELO
      SINANG = VCOMPR/QVELO
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
      CALL E2VECT(ALVECT)

C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C     IS EMANATING

      PHYPO  = (SONDPR - QVELO)*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) + PHYPO*COSANG
      YPNT   = GEOMG2(2,INODE) + PHYPO*SINANG
C
C     FIND IN WHICH CELL THE POINT (XPNT,YPNT) IS LOCATED AND
C     INTERPOLATE IN CARTESIAN COORDINATES AT THIS POINT STORING
C     VALUES IN DPENJA(*)
C
      CALL G2LCAT (IBOUND, XPNT, YPNT)
C
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AGAIN
C
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
C
C     COMPUTE SOURCE TERMS AT THE INTERIOR POINT (ROTATED SYSTEM)
C
      CALL FRSOUR
C
C     COMPUTE THE PRODUCTS LU AND LW
C
      EIGENU(1) = 0.
      EIGENW(1) = 0.

      DO 510 JQ = 1, NEQNFL
        EIGENU(1) = EIGENU(1) + ALVECT(1,JQ)*DPENJA(JQ)
        EIGENW(1) = EIGENW(1) + ALVECT(1,JQ)*BIGWJA(JQ)
510   CONTINUE

      EIGENU(1) = EIGENU(1) + EIGENW(1)*CELLTI(IONE)
C
C     NOW COMPUTE THE CHARACTERISTICS FROM EXTERIOR DOMAIN
C     FIRST SET THE DEPENDENT VARIABLES
C
      DO 515 IQ = 1, NEQNFL
         DPENJA(IQ) = DPENFR(IQ)
515   CONTINUE
C
```

```
C         ROTATE THESE VALUES ALONG THE STREAMLINE AT THE SAME ANGLE
C
          DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
          DPENJA(3) = 0.
C
C         NOW COMPUTE THE MATRIX PRODUCTS
C
          DO 525 IQ = 2, NEQNFL
            EIGENU(IQ) = 0.
            DO 520 JQ = 1, NEQNFL
              EIGENU(IQ) = EIGENU(IQ) + ALVECT(IQ,JQ)*DPENJA(JQ)
520         CONTINUE
525       CONTINUE
C
C         NOW INVERT THIS TO COMPUTE THE REAL VALUES
C
          CALL GAUSS2 (ALVECT,EIGENU,DPENJA,NEQNFL,MEQNFL)
C
C         RECOMPUTE THE VELOCITY SO THAT IT CAN BE DISTRIBUTED
C
          UCOMPR    = DPENJA(2)/DPENJA(1)
          VCOMPR    = DPENJA(3)/DPENJA(1)
          QVELO     = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
          COSNEW    = UCOMPR/QVELO
          SINNEW    = VCOMPR/QVELO
          SINDUM    = SINANG*COSNEW+COSANG*SINNEW
          COSANG    = COSANG*COSNEW-SINANG*SINNEW
          SINANG    = SINDUM
          UCOMPR    = QVELO*COSANG
          VCOMPR    = QVELO*SINANG
          DPENJA(2) = UCOMPR*DPENJA(1)
          DPENJA(3) = VCOMPR*DPENJA(1)

          DO 530 JQ = 1, NEQNFL
            CHNGE2(JQ,INODE) = DPENJA(JQ) - DPENG2(JQ,INODE)
530       CONTINUE
C
          GO TO 1500
C
C         ----------------------
C         CHARACTERISTIC OUTFLOW
C         ----------------------
C
C         REVISE THE FOLLOWING ONCE IT STARTS WORKING
C
600       IF (ITWO .NE. 0) GOTO 604
          IF (IEDGE .EQ. 4) THEN
            DO 602 IQ = 1, NEQNFL
              DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(6,IONE))
              CHNGE2(IQ,INODE) = 0.
602         CONTINUE
            GOTO 1500
          ENDIF
C
          IF (IEDGE .EQ. 6) THEN
            DO 603 IQ = 1, NEQNFL
              DPENG2(IQ,INODE) = DPENG2(IQ,ICELG2(4,IONE))
```

```
            CHNGE2(IQ,INODE) = 0.
603         CONTINUE
            GOTO 1500
          ENDIF
C
C         SET UP THE DEPENDENT VARIABLES
C
604       IFAC = 1
          DO 605 IQ = 1, NEQNFL
             DPENJA(IQ) = DPENG2(IQ,INODE) + IFAC*CHNGE2(IQ,INODE)
             DPENSV(IQ) = DPENJA(IQ)
605       CONTINUE
C
          IF (IEDGE .EQ. 3 .AND. DPENJA(3) .GT. 0.) THEN
              DPENJA(3)       = 0.
              DPENG2(3,INODE) = 0.
              CHNGE2(3,INODE) = 0.
          ENDIF
C
C         COMPUTE THE VELOCITY COMPONENTS, PRESSURE, GAMMA ETC
C
          CALL FLBGF2
C
          QVELO  = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
          SONDPR = SQRT(GAMAPR*PRESPR/RHORPR)

C         DETERMINE IF SUPERSONIC EXIT
C
          IF (QVELO .GT. SONDPR) GO TO 700
C
C         ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AT
C         THE BOUNDARY NODE AND COMPUTE EIGENVECTOR MATRIX IN THIS
C         ROTATED SYSTEM

          COSANG    = UCOMPR/QVELO
          SINANG    = VCOMPR/QVELO
          DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
          DPENJA(3) = 0.
C         DPENF2    = QVELO*DPENFR(1)
          CALL E2VECT(ALVECT)
C
C         DETERMINE THE SOURCE TERMS AT THE BOUNDARY NODE; SAVE THEM
C
          CALL FRSOUR
          DO 610 IQ = 1, NEQNFL
             BIGWSV(IQ) = BIGWJA(IQ)
610       CONTINUE
C
C         DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C         WITH SPEED (U+A) IS EMANATING

          PHYPO  = (SONDPR + QVELO)*CELLTI(IONE)
          XPNT   = GEOMG2(1,INODE) - PHYPO*COSANG
          YPNT   = GEOMG2(2,INODE) - PHYPO*SINANG
C
C         FIND IN WHICH CELL THE POINT (XPNT,YPNT) IS LOCATED AND
C         INTERPOLATE IN CARTESIAN COORDINATES AT THIS POINT STORING
```

```
C         VALUES IN DPENJA(*)
C
          CALL G2LCAT (IBOUND, XPNT, YPNT)
C
C         ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AGAIN
C
          DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
          DPENJA(3) = 0.
          DPENSV(2) = DPENSV(2)*COSANG + DPENSV(3)*SINANG
          DPENSV(3) = 0.
C
C         COMPUTE SOURCE TERMS AT THE INTERIOR POINT (ROTATED SYSTEM)
C
          CALL FRSOUR
C
C         COMPUTE THE PRODUCTS LU AND LW
C
          EIGENU(2) = 0.
          EIGENW(2) = 0.

          DO 615 JQ = 1, NEQNFL
            EIGENU(2) = EIGENU(2) + ALVECT(2,JQ)*DPENJA(JQ)
            EIGENW(2) = EIGENW(2) + ALVECT(2,JQ)*BIGWJA(JQ)
615       CONTINUE

C         DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C         WITH SPEEDS U ARE EMANATING

          PHYPON = QVELO*CELLTI(IONE)
          XPNT   = GEOMG2(1,INODE) - PHYPON*COSANG
          YPNT   = GEOMG2(2,INODE) - PHYPON*SINANG
C
C         INTERPOLATE VALUES
C
          XL        = PHYPON/PHYPO
          XLM1      = 1. - XL
C
          DO 620 IQ = 1, NEQNFL
            DPENJA(IQ) = DPENJA(IQ)*XL + DPENSV(IQ)*XLM1
            BIGWJA(IQ) = BIGWJA(IQ)*XL + BIGWSV(IQ)*XLM1
620       CONTINUE
C
C         NOW COMPUTE THE MATRIX PRODUCTS
C
          DO 630 IQ = 3, NEQNFL
            EIGENU(IQ) = 0.
            EIGENW(IQ) = 0.
            DO 625 JQ = 1, NEQNFL
              EIGENU(IQ) = EIGENU(IQ) + ALVECT(IQ,JQ)*DPENJA(JQ)
              EIGENW(IQ) = EIGENW(IQ) + ALVECT(IQ,JQ)*BIGWJA(JQ)
625         CONTINUE
630       CONTINUE
C
C         CORRECT THE INTERIOR CHARACTERISTICS FOR TIME
C
          DO 635 IQ = 2, NEQNFL
            EIGENU(IQ) = EIGENU(IQ) + EIGENW(IQ)*CELLTI(IONE)
```

```
635       CONTINUE
C
C         NOW COMPUTE THE CHARACTERISTICS FROM EXTERIOR DOMAIN
C         FIRST SET THE DEPENDENT VARIABLES
C
          DO 640 IQ = 1, NEQNFL
             DPENJA(IQ) = DPENFR(IQ)
640       CONTINUE
C
C         ROTATE THESE VALUES ALONG THE STREAMLINE AT THE SAME ANGLE
C
          DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
C         DPENJA(2) = DPENF2
          DPENJA(3) = 0.

          EIGENU(1) = 0.
          DO 645 IQ = 1, NEQNFL
             EIGENU(1) = EIGENU(1) + ALVECT(1,IQ)*DPENJA(IQ)
645       CONTINUE
C
C         NOW INVERT THIS TO COMPUTE THE REAL VALUES
C
          CALL GAUSS2 (ALVECT,EIGENU,DPENJA,NEQNFL,MEQNFL)
C
C         RECOMPUTE THE VELOCITY SO THAT IT CAN BE DISTRIBUTED
C
          UCOMPR    = DPENJA(2)/DPENJA(1)
          VCOMPR    = DPENJA(3)/DPENJA(1)
          QVELO     = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
          UCOMPR    = QVELO*COSANG
          VCOMPR    = QVELO*SINANG
          DPENJA(2) = UCOMPR*DPENJA(1)
          DPENJA(3) = VCOMPR*DPENJA(1)

          DO 650 JQ = 1, NEQNFL
            CHNGE2(JQ,INODE) = DPENJA(JQ) - DPENG2(JQ,INODE)
650       CONTINUE
C
          GO TO 1500
C
C         ---------------------------------
C         CHARACTERISTIC SUPERSONIC OUTFLOW
C         ---------------------------------
C
C         DETERMINE THE SOURCE TERMS AT THE BOUNDARY NODE; SAVE THEM
C
700       CALL FRSOUR
          DO 705 IQ = 1, NEQNFL
             BIGWSV(IQ) = BIGWJA(IQ)
705       CONTINUE
C
C         ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AT
C         THE BOUNDARY NODE AND COMPUTE EIGENVECTOR MATRIX IN THIS
C         ROTATED SYSTEM
C
          COSANG = UCOMPR/QVELO
          SINANG = VCOMPR/QVELO
```

```
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
      CALL E2VECT(ALVECT)
C
C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C     (U+A) IS EMANATING

      PHYPO  = (SONDPR + QVELO)*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) - PHYPO*COSANG
      YPNT   = GEOMG2(2,INODE) - PHYPO*SINANG
C
C     FIND IN WHICH CELL THE POINT (XPNT,YPNT) IS LOCATED AND
C     INTERPOLATE IN CARTESIAN COORDINATES AT THIS POINT STORING
C     VALUES IN DPENJA(*)
C
      CALL G2LCAT (IBOUND, XPNT, YPNT)
C
C     THE DIRECTION OF THE STREAM LINE MIGHT HAVE CHANGED, SO
C     CORRECT IT
C     ROTATE THE DEPENDENT VARIABLES (NATURAL COORDINATES) AGAIN
C
      DPENJA(2) = DPENJA(2)*COSANG + DPENJA(3)*SINANG
      DPENJA(3) = 0.
      DPENSV(2) = DPENSV(2)*COSANG + DPENSV(3)*SINANG
      DPENSV(3) = 0.
C
C     COMPUTE SOURCE TERMS AT THE INTERIOR POINT (ROTATED SYSTEM)
C
      CALL FRSOUR
C
C     COMPUTE THE PRODUCTS LU AND LW
C
      EIGENU(2) = 0.
      EIGENW(2) = 0.

      DO 710 JQ = 1, NEQNFL
        EIGENU(2) = EIGENU(2) + ALVECT(2,JQ)*DPENJA(JQ)
        EIGENW(2) = EIGENW(2) + ALVECT(2,JQ)*BIGWJA(JQ)
710   CONTINUE
C
C     DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTICS
C     WITH SPEEDS U ARE EMANATING
C
      PHYPON = QVELO*CELLTI(IONE)
      XPNT   = GEOMG2(1,INODE) - PHYPON*COSANG
      YPNT   = GEOMG2(2,INODE) - PHYPON*SINANG
C
C     INTERPOLATE VALUES
C
      XL    = PHYPON/PHYPO
      XLM1  = 1. - XL
      PHYPO = PHYPON
C
      DO 715 IQ = 1, NEQNFL
        DPENJA(IQ) = DPENJA(IQ)*XL + DPENSV(IQ)*XLM1
        BIGWJA(IQ) = BIGWJA(IQ)*XL + BIGWSV(IQ)*XLM1
715   CONTINUE
```

```
C
C         NOW COMPUTE THE MATRIX PRODUCTS
C
          DO 725 IQ = 3, NEQNFL
            EIGENU(IQ) = 0.
            EIGENW(IQ) = 0.
            DO 720 JQ = 1, NEQNFL
              EIGENU(IQ) = EIGENU(IQ) + ALVECT(IQ,JQ)*DPENJA(JQ)
              EIGENW(IQ) = EIGENW(IQ) + ALVECT(IQ,JQ)*BIGWJA(JQ)
720         CONTINUE
725       CONTINUE
C
C         DETERMINE THE DISTANCE FROM WHERE THE INTERIOR CHARACTERISTIC
C         WITH SPEED (U-A) IS EMANATING

          PHYPON = (QVELO - SONDPR)*CELLTI(IONE)
          XPNT   = GEOMG2(1,INODE) - PHYPON*COSANG
          YPNT   = GEOMG2(2,INODE) - PHYPON*SINANG
C
C         INTERPOLATE VALUES
C
          XL         = PHYPON/PHYPO
          XLM1       = 1. - XL
C
          DO 730 IQ = 1, NEQNFL
            DPENJA(IQ) = DPENJA(IQ)*XL + DPENSV(IQ)*XLM1
            BIGWJA(IQ) = BIGWJA(IQ)*XL + BIGWSV(IQ)*XLM1
730       CONTINUE
C
          EIGENU(1) = 0.
          EIGENW(1) = 0.
          DO 735 IQ = 1, NEQNFL
             EIGENU(1) = EIGENU(1) + ALVECT(1,IQ)*DPENJA(IQ)
             EIGENW(1) = EIGENW(1) + ALVECT(1,IQ)*BIGWJA(IQ)
735       CONTINUE
C
C         CORRECT THE INTERIOR CHARACTERISTICS FOR TIME
C
          DO 740 IQ = 1, NEQNFL
            EIGENU(IQ) = EIGENU(IQ) + EIGENW(IQ)*CELLTI(IONE)
740       CONTINUE
C
C         NOW INVERT THIS TO COMPUTE THE REAL VALUES
C
          CALL GAUSS2 (ALVECT,EIGENU,DPENJA,NEQNFL,MEQNFL)
C
C         RECOMPUTE THE VELOCITY SO THAT IT CAN BE DISTRIBUTED
C
          UCOMPR     = DPENJA(2)/DPENJA(1)
          VCOMPR     = DPENJA(3)/DPENJA(1)
          QVELO      = SQRT(UCOMPR*UCOMPR + VCOMPR*VCOMPR)
          UCOMPR     = QVELO*COSANG
          VCOMPR     = QVELO*SINANG
          DPENJA(2) = UCOMPR*DPENJA(1)
          DPENJA(3) = VCOMPR*DPENJA(1)

          DO 745 JQ = 1, NEQNFL
```

```
          CHNGE2(JQ,INODE) = DPENJA(JQ) - DPENG2(JQ,INODE)
745       CONTINUE

          GOTO 1500
C
C         --------------------
C         EQUILIBRIUM BOUNDARY
C         --------------------
C
800       CONTINUE

          JFAC   = 2
          IF (ITWO .EQ. 0) JFAC = 2*JFAC

810       RHORPR = DPENG2(1,INODE)
          RECDEN = 1./RHORPR
          UCOMPR = DPENG2(2,INODE)*RECDEN
          VCOMPR = DPENG2(3,INODE)*RECDEN
          VELO2U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C         COMPUTE THE DIMENSIONAL QUANTITIES
          TEMPPR = TEMPG2(INODE)*TREFFL
C
C         SET THE FIRST FOUR CHANGES AS ZERO'S
C
          CHNGE2(1,INODE) = 0.
          CHNGE2(2,INODE) = 0.
          CHNGE2(3,INODE) = 0.
          CHNGE2(4,INODE) = 0.

C         COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
          SUMY   = 0.
          YUPPER = 1. - YNRTCH

          DO 820 JS = NEQBAS+1, NEQNFL
             IS          = JS - NEQBAS
             DPENG2(JS,INODE) = DPENG2(JS,INODE) +JFAC*CHNGE2(JS,INODE)
             CHNGE2(JS,INODE) = 0.
             YSPEPR(IS) = DPENG2(JS,INODE)*RECDEN
             IF (YSPEPR(IS) .LT. 0.) THEN
                YSPEPR(IS)       = 0.
                DPENG2(JS,INODE) = 0.
             ENDIF
C            IF (YSPEPR(IS) .GT. YUPPER) THEN
C               YSPEPR(IS)       = YUPPER
C               DPENG2(JS,INODE) = YUPPER*DPENG2(1,INODE)
C            ENDIF
             SUMY       = SUMY + YSPEPR(IS)
820       CONTINUE

          YSPEPR(NEQSCH+1) = YUPPER - SUMY
C         YSPEPR(NEQSCH+1) = ABS(1. - SUMY - YNRTCH)
          IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C
C         COMPUTE THE ENTHALPY OF THE MIXTURE
C
          SYSHFS = 0.
          SYSCPS = 0.
```

```
          SYSBMS = 0.
          BIGAM  = 0.

          DO 830 IS = 1, NSPECH
             SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
             SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
             SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
             BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
830       CONTINUE

          ENTHAL = SYSHFS + SYSCPS*(TEMPPR-TREFCH) +
     1             0.5*BIGAM*(TEMPPR**2-TREFCH**2)
          ENTHAL = ENTHAL/FMREFL + 0.5*VEL02U

C         COMPUTE THE DIMENSIONLESS PRESSURE
C
          PRESG2(INODE) = RHORPR*TEMPG2(INODE)*AMWTFL*SYSBMS
C
C         COMPUTE THE FOURTH COMPONENT OF STATE VECTOR
C
          DPENG2(4,INODE) = RHORPR*ENTHAL - PRESG2(INODE)

          GOTO 1500
C
C         -------------------------
C         CORNER BOUNDARY CONDITIONS
C         -------------------------
C
900       CONTINUE

c         write(6,*) ' it does come here e2bcn0 iedge ',iedge,inode
          IF (IEDGE .EQ. 2) THEN
             INEXTN = 8
          ELSE IF (IEDGE .EQ. 4) THEN
             INEXTN = 6
          ELSE IF (IEDGE .EQ. 6) THEN
             INEXTN = 4
          ELSE IF (IEDGE .EQ. 8) THEN
             INEXTN = 2
          ELSE
             GOTO 1500
          ENDIF
          NODENB = ICELG2(INEXTN,IONE)
c
          DO 910 JS = 1, NEQNFL
             DPENG2(JS,INODE) = DPENG2(JS,NODENB)
             CHNGE2(JS,INODE) = 0.
910       CONTINUE
          GOTO 1500
C
C         ----------------------------
C         RADIATION CONDITION + FACTOR 2
C         ----------------------------
C
C         SUPERSONIC EXIT
C
1000      DO 1010 IQ = 1, NEQNFL
```

```
            CHNGE2(IQ,INODE) = 2.*CHNGE2(IQ,INODE)
1010      CONTINUE

          GO TO 1500
C
C         ---------------------
C         VISCOUS WALL BOUNDARY
C         ---------------------
C
C1100     IFACTR = 2
1100      FFACTR = 2.

          GO TO (1105,1110,1105,1110,1105,1110,1105,1110), (IEDGE-1)
          GO TO 1500

C1105     IFACTR = 4
1105      FFACTR = 4.
1110      IF (IEDGE .EQ. 0) FFACTR = 4./3.

C         SET THE VELOCITIES ZERO AND OTHER VALUES AS REFLECTION
          DO 1120 IQ = 1, NEQNFL
C           CHNGE2(IQ,INODE) = IFACTR*CHNGE2(IQ,INODE)
            CHNGE2(IQ,INODE) = FFACTR*CHNGE2(IQ,INODE)
1120      CONTINUE

          CHNGE2(2,INODE) = 0.
          CHNGE2(3,INODE) = 0.
          DPENG2(2,INODE) = 0.
          DPENG2(3,INODE) = 0.

C         GO TO 1500
C
C         PRINT OUT PARAMETERS
C
1500      IF (IWRITE) THEN
               WRITE(JDEBUG,2000)
               WRITE(JDEBUG,2100)
               WRITE(JDEBUG,2200)
               WRITE(JDEBUG,2300) INODE,IONE,ITWO,IEDGE,ITYPE,ITGL
               WRITE(JDEBUG,2400)
               WRITE(JDEBUG,2600) (DPENG2(IQ,INODE), IQ=1,NEQNFL)
               WRITE(JDEBUG,2500)
               WRITE(JDEBUG,2600) (CHNGE2(IQ,INODE), IQ=1,NEQNFL)
               IF (ITYPE .EQ. 3) THEN
                 WRITE(JDEBUG,2700) KNODE1,KNODE2,DXSIDE,DYSIDE,
     1              COSANG,SINANG,RHOU,RHOV,RHO,RHOQ,U2NEXT,U3NEXT
               ENDIF
          ENDIF

C         GO BACK FOR NEXT NODE

1600    CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------
```

```
2000    FORMAT(//10X,'----------------------' )
2100    FORMAT(  10X,'DEBUG PRINT FROM E2BCNO' )
2200    FORMAT(  10X,'----------------------'/)
2300    FORMAT(5X,'INODE = ',I5,10X,'IONE  = ',I5,10X,'ITWO = ',I5/
     1         5X,'IEDGE = ',I5,10X,'ITYPE = ',I5,10X,'ITGL = ',I5/)
2400    FORMAT(/5X,'DEPENDENT VARIABLES')
2500    FORMAT(/5X,'CHANGE VARIABLES')
2600    FORMAT (8G14.5)
2700    FORMAT(5X, 'KNODE1=',I5,  10X, 'KNODE2=',I5,
     2        15X, 'DXSIDE=',F10.5,5X, 'DYSIDE=',F10.5/
     3         5X, 'COSANG=',F10.5,5X, 'SINANG=',F10.5,
     4        10X, 'RHOU  =',F10.5,5X, 'RHOV  =',F10.5/
     5         5X, 'RHO   =',F10.5,5X, 'RHOQ  =',F10.5,
     6        10X, 'U2NEXT=',F10.5,5X, 'U3NEXT=',F10.5)

c       IF (IADDH2 .NE. 0) THEN
c         WRITE(6,*) ' HEYMAN MAN MUMDH2 IN E2BCNO',numdh2
c         CALL H2MIXT(ITGL)
c         do j = 1, neqnfl
c           chnge2(j,21) = 0.
c           chnge2(j,22) = 0.
c         enddo
c       ENDIF
        RETURN
        END
```

E2CON0

```
        SUBROUTINE E2CON0 (TIME, ITGL, IPASS, IPASSM)

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'TICOMN.INC'

        DIMENSION ERROR(3)
        SAVE ERROR, FACTOR

C***********************************************************************

C       THIS SUBROUTINE COMPUTES THE CONVERGENCE HISTORY.
C       THE ERRORS ARE COMPUTED FOR THE VARIABLE (EQUATION) KEQNE2,
C       THUS, E.G., KEQNE2 = 2 FOR THE MOMENTUM CONSERVATION.
C       THE ERROR TYPE CALCULATION IS STORED IN ERORE2, AND ITS TYPE
C       IS DETERMINED BY THE VARIABLE KONVE2, WHICH CAN HAVE THE
C       FOLLOWING VALUES:
C           KONVE2 = 1      ===>  AVERAGE ERROR = ERROR(1)
C           KONVE2 = 2      ===>  MAXIMUM ERROR = ERROR(2)
C           KONVE2 = 3      ===>  RMS     ERROR = ERROR(3)
```

```
C*********************************************************************
C       INITIALIZE THE ERRORS

        ERORE2   = 100.
        ERRORM   = 0.
        FACTMN   = 0.125

        IF (IPASS .EQ. 1) THEN
          ERROR(1) = 0.
          ERROR(2) = 0.
          ERROR(3) = 0.
          FACTOR   = 1.
          JEQM     = 0
          FCTRTI   = MAX (FCTRTI, FACTMN)
        ENDIF

C       DETERMINE IF THE CELL TIME-STEPS ARE TO BE ADJUSTED

        IF (KFACTI .EQ. 0) GOTO 30

        DO 20 JNODE = ILVLA2(1,ITGL),ILVLA2(2,ITGL)
          INODE = MRKDA2(JNODE)
          DO 10 JQ = 1, NEQNFL
            ABSERR = ABS(CHNGE2(JQ,INODE))
            IF (ABSERR .GT. ERRORM) THEN
              ERRORM = ABSERR
              JEQM   = JQ
            ENDIF
10        CONTINUE
20      CONTINUE

        IF (ERRORM .GT. ERRMTI) THEN
           FACTOR = ERRMTI/ERRORM
           IF (DTCNTI .GT. 0.) THEN
             IF (FACTOR .GT. 0.5) THEN
                FACTOR = 0.5
             ELSEIF (FACTOR .GT. 0.25) THEN
                  FACTOR = 0.25
             ELSE
                  FACTOR = 0.125
             ENDIF
           ENDIF
           FACTOR = MAX (FACTOR, FACTMN)
        ENDIF

        FCTRTI = MIN (FCTRTI, FACTOR)

C       SEE IF YOU WANT TO REALLY COLLECT CONVERGENCE HISTORY ?

30      IF (KONVE2 .EQ. 0) THEN
          IF (IPASS .EQ. IPASSM) THEN
            WRITE (JTERMO,1000) NITRE2,JEQM,FCTRTI,ERRORM,DTMNTI,TIME
          ENDIF
          RETURN
        ENDIF
```

```
C       LOOP OVER ALL THE NODES AT THIS LEVEL AND COLLECT ERRORS

        DO 50 JNODE = ILVLA2(1,ITGL), ILVLA2(2,ITGL)

C           ACTUAL NODE ASSIGNMENTS

            INODE = MRKDA2 (  JNODE)

            ABSERR      = ABS(CHNGE2(KEQNE2,INODE))
            ERROR(1)    = ERROR(1) + ABSERR
            ERROR(2)    = MAX (ERROR(2),ABSERR)
            ERROR(3)    = ERROR(3) + ABSERR*ABSERR
            IF (ERROR(2) .EQ. ABSERR) THEN
               IP = INODE
               ERRMAX = CHNGE2(KEQNE2,INODE)
            ENDIF

50      CONTINUE

        IF (IPASS .EQ. IPASSM) THEN
           ERROR(1) = ERROR(1)/FLOAT(NNODA2)
           ERROR(3) = SQRT(ERROR(3)/FLOAT(NNODA2))
           ERORE2   = ERROR(KONVE2)

C          WRITE THE FOLLOWING :
C             1: NITRE2   : ITERATION COUNTER
C             2: KONVE2   : TYPE OF ERROR
C             3: IP       : POSITION OF MAXIMUM ERROR
C             4: ERROR(1) : AVERAGE ERROR
C             5: ERROR(2) : MAXIMUM ERROR
C             6: ERROR(3) : RMS     ERROR

           WRITE (JHISTO,1100) NITRE2  , IP      , KONVE2  , KEQNE2,
     1                         ERROR(1), ERROR(2), ERROR(3), TIME
           WRITE (JTERMO,1100) NITRE2  , IP      , KONVE2  , KEQNE2,
     1                         ERROR(1), ERRMAX  , ERROR(3), TIME

        ENDIF
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(I5,2X,I2,3X,4G15.5)
1100    FORMAT(2I5,1X,I2,1X,I2,2X,4G15.5)

        RETURN
        END
```

E2CONF

```
        SUBROUTINE E2CONO (TIME, ITGL, IPASS, IPASSM)
```

```
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'TICOMN.INC'

      DIMENSION ERROR(3)
      SAVE ERROR

C*********************************************************************

C     THIS SUBROUTINE COMPUTES THE CONVERGENCE HISTORY.
C     THE ERRORS ARE COMPUTED FOR THE VARIABLE (EQUATION) KEQNE2,
C     THUS, E.G., KEQNE2 = 2 FOR THE MOMENTUM CONSERVATION.
C     THE ERROR TYPE CALCULATION IS STORED IN ERORE2, AND ITS TYPE
C     IS DETERMINED BY THE VARIABLE KONVE2, WHICH CAN HAVE THE
C     FOLLOWING VALUES:
C         KONVE2 = 1     ===>  AVERAGE ERROR = ERROR(1)
C         KONVE2 = 2     ===>  MAXIMUM ERROR = ERROR(2)
C         KONVE2 = 3     ===>  RMS     ERROR = ERROR(3)

C*********************************************************************

C     INITIALIZE THE ERRORS

      ERORE2   = 100.
      ERRORM   = 0.
      FCTRTI   = 1.

      IF (IPASS .EQ. 1) THEN
        ERROR(1) = 0.
        ERROR(2) = 0.
        ERROR(3) = 0.
        JEQM     = 0
      ENDIF

C     DETERMINE IF THE CELL TIME-STEPS ARE TO BE ADJUSTED

      IF (KFACTI .EQ. 0) GOTO 30

      DO 20 JNODE = ILVLA2(1,ITGL),ILVLA2(2,ITGL)
        INODE = MRKDA2(JNODE)
        DO 10 JQ = 1, NEQNFL
          ABSERR = ABS(CHNGE2(JQ,INODE))
          IF (ABSERR .GT. ERRORM) THEN
            ERRORM = ABSERR
            JEQM   = JQ
          ENDIF
10      CONTINUE
20    CONTINUE

C     SEE IF YOU WANT TO REALLY COLLECT CONVERGENCE HISTORY ?

30    IF (KONVE2 .EQ. 0) THEN
        IF (IPASS .EQ. IPASSM) THEN
```

```
            WRITE (JTERMO,1000) NITRE2,JEQM,FCTRTI,ERRORM,DTMNTI,TIME
          ENDIF
          RETURN
        ENDIF

C       LOOP OVER ALL THE NODES AT THIS LEVEL AND COLLECT ERRORS

        DO 50 JNODE = ILVLA2(1,ITGL), ILVLA2(2,ITGL)

C           ACTUAL NODE ASSIGNMENTS

            INODE = MRKDA2 (  JNODE)

            ABSERR      = ABS(CHNGE2(KEQNE2,INODE))
            ERROR(1)    = ERROR(1) + ABSERR
            ERROR(2)    = MAX (ERROR(2),ABSERR)
            ERROR(3)    = ERROR(3) + ABSERR*ABSERR
            IF (ERROR(2) .EQ. ABSERR) THEN
               IP = INODE
               ERRMAX = CHNGE2(KEQNE2,INODE)
            ENDIF

50      CONTINUE

        IF (IPASS .EQ. IPASSM) THEN
           ERROR(1) = ERROR(1)/FLOAT(NNODA2)
           ERROR(3) = SQRT(ERROR(3)/FLOAT(NNODA2))
           ERORE2   = ERROR(KONVE2)

C          WRITE THE FOLLOWING :
C             1: NITRE2   : ITERATION COUNTER
C             2: KONVE2   : TYPE OF ERROR
C             3: IP       : POSITION OF MAXIMUM ERROR
C             4: ERROR(1) : AVERAGE ERROR
C             5: ERROR(2) : MAXIMUM ERROR
C             6: ERROR(3) : RMS     ERROR

           WRITE (JHISTO,1100) NITRE2  , IP      , KONVE2  , KEQNE2,
     1                         ERROR(1), ERROR(2), ERROR(3), TIME
           WRITE (JTERMO,1100) NITRE2  , IP      , KONVE2  , KEQNE2,
     1                         ERROR(1), ERRMAX  , ERROR(3), TIME

        ENDIF
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(I5,2X,I2,3X,4G15.5)
1100    FORMAT(2I5,1X,I2,1X,I2,2X,4G15.5)

        RETURN
        END
```

E2CORB

```
      SUBROUTINE E2CORB

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] SPCOMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE APPLIES THE BOUNDARY CONDITIONS AT THE SPECIAL
C     CORNER NODES.  THE VALUES ASSIGNED TO THESE NODES ARE THE VALUES
C     OF SOME NEIGHBOURING NODE

C*********************************************************************
C
      DO 20 IBOUND = 1, LBNDG2

C       BRANCH OUT ACCORDING TO TYPE
C       INODE IS THE BOUNDARY NODE
C       NBP  IS THE NEIGHBOUR NODE POINTER OF THE ADJACENT CELL
C       ICP  IS THE NODE POINTER OF THE ADJACENT CELL

        INODE  = JBNDG2(1,IBOUND)
        NBP    = JBNDG2(2,IBOUND)
        ICP    = JBNDG2(3,IBOUND)
        NBCELL = NEIBG2(NBP,INODE)
        INEXT  = ICELG2(ICP,NBCELL)
        DO 10 IEQ = 1, NEQNFL
           DPENG2(IEQ,INODE) = DPENG2(IEQ,INEXT)
           CHNGE2(IEQ,INODE) = 0.
10      CONTINUE
20    CONTINUE

      RETURN
      END
```

E2CORF

```
      SUBROUTINE E2CORF

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] SPCOMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE INITIALIZES THE SPECIAL BOUNDARY CONDITIONS
C     POINTERS FOR THE CORNER NODES
```

```
C*********************************************************************
C
      DO 20 IBOUND = 1, LBNDG2
        IBNDG2(1,NBNDG2+IBOUND) = JBNDG2(1,IBOUND)
        IBNDG2(2,NBNDG2+IBOUND) = JBNDG2(2,IBOUND)
        IBNDG2(3,NBNDG2+IBOUND) = JBNDG2(3,IBOUND)
        IBNDG2(5,NBNDG2+IBOUND) = 99
20    CONTINUE

      NBNDG2 = NBNDG2 + LBNDG2

      RETURN
      END
```

E2CORI

```
      SUBROUTINE E2CORI

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] SPCOMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE INITIALIZES THE SPECIAL BOUNDARY CONDITIONS
C     POINTERS FOR THE CORNER NODES

C*********************************************************************
C
C
      LBNDG2 = 0
      DO 20 IBOUND = 1, NBNDG2

         ITYPE = IBNDG2(5,IBOUND)

         IF (ITYPE .EQ. 99) THEN
            LBNDG2           = LBNDG2 + 1
            JBNDG2(1,LBNDG2) = IBNDG2(1,IBOUND)
            JBNDG2(2,LBNDG2) = IBNDG2(2,IBOUND)
            JBNDG2(3,LBNDG2) = IBNDG2(3,IBOUND)
            IBNDG2(1,IBOUND) = 0
            IBNDG2(2,IBOUND) = 0
            IBNDG2(3,IBOUND) = 0
            IBNDG2(4,IBOUND) = 0
            IBNDG2(5,IBOUND) = 0
         ENDIF

20    CONTINUE

      NBNDG2 = NBNDG2 - LBNDG2
```

```
      RETURN
      END _
```

E2DIFF

```
      SUBROUTINE E2DIFF

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      DATA KOUNT /0/

C*********************************************************************

C     THIS SUBROUTINE STEPS THROUGH EACH CEWIC CELL AND COMPUTES THE
C     ARTIFICIAL VISCOSITY COEFFICIENT

C*********************************************************************

C     SEE IF YOU WANT TO USE A CONSTANT VISOCITY MODEL

      IF (SMAXE2 .LE. SMINE2) RETURN

      DSIGMX = 0.

C     INITIALIZE THE ARTIFICIAL VISCOSITY AT EACH NODE

      DO 10 IN = 1, NNODG2
        SIGGE2(IN) = SMINE2
10    CONTINUE

C     STEP THROUGH EACH CEWIC CELL

      DO 20 JCELL = 1, NCELA2

C       FIND THE ACTUAL CELL NUMBER

        ICELL = ICELA2(JCELL)

C       SET UP NODE POINTERS FOR THIS CELL

        KSW = ICELG2(2,ICELL)
        KSE = ICELG2(4,ICELL)
        KNE = ICELG2(6,ICELL)
        KNW = ICELG2(8,ICELL)

C       STORE DENSITY AT THE FOUR NODES AND EDGES

        RSW = DPENG2(1,KSW)
        RSE = DPENG2(1,KSE)
```

```
            RNE = DPENG2(1,KNE)
            RNW = DPENG2(1,KNW)

            RE  = RNE + RSE
            RW  = RNW + RSW
            RN  = RNE + RNW
            RS  = RSE + RSW

C           COMPUTE THE VISCOSITY COEFFICIENT BASED UPON GRADIENTS
C           NOTE THAT THE FOLLOWING MODEL WILL SET MAXIMUM VISCOSITY
C           AT THE NODE WHERE DSIGR IS MAXIMUM

            DSIGRX      = ABS(RE-RW)/(RE+RW)
            DSIGRY      = ABS(RN-RS)/(RN+RS)
            DSIGT       = DSIGRX + DSIGRY
            DSIGMX      = MAX (DSIGT, DSIGMX)
            SIGNOD      = 0.25*SDELE2*DSIGT
            SIGGE2(KSW) = SIGGE2(KSW) + SIGNOD
            SIGGE2(KSE) = SIGGE2(KSE) + SIGNOD
            SIGGE2(KNE) = SIGGE2(KNE) + SIGNOD
            SIGGE2(KNW) = SIGGE2(KNW) + SIGNOD
C
C           THE FOLLOWING IS NEEDED ONLY FOR DEBUG PURPOSE
C
            IF (DSIGMX .EQ. DSIGT) ICELSM = ICELL

20      CONTINUE

C       ADJUST THE PARAMETER MULTIPLYING THE COEFFICIENTS

        IF (DSIGMX .NE. 0.) SDELE2 = (SMAXE2-SMINE2)/DSIGMX

C       CORRECT THE ARTIFICIAL VISCOSITY AT THE BOUNDARIES
C
        DO 30 IN = 1, NBNDG2
           INODE = IBNDG2(1,IN)
           SIGGE2(INODE) = 2.*SIGGE2(INODE) - SMINE2
30      CONTINUE
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGE2 .NE. 3 .AND. IDBGE2 .LT. 1000)  RETURN

        IF (KOUNT .EQ. 0) THEN
           KOUNT = 1
           WRITE(JDEBUG,1000)
           WRITE(JDEBUG,1100)
           WRITE(JDEBUG,1200)
        ENDIF

        WRITE(JDEBUG,1300) ICELSM, SMAXE2, SMINE2, DSIGMX, SDELE2

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
```

```
1100    FORMAT(  10X,'DEBUG PRINT FROM E2DIFF' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'CELL OF MAXIMUM VISCOCITY COEFFICIENT =',I5/
     1         5X,'SMAXE2=',G14.5,5X,'SMINE2=',G14.5,
     2         5X,'DSIGMX=',G14.5,5X,'SDELE2=',G14.5          )

        RETURN
        END
```

E2FINI

```
        SUBROUTINE E2FINI
C
        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] A2COMN.INC/LIST'
        INCLUDE '[.INC] E2COMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        INCLUDE '[.INC] KYCOMN.INC/LIST'
        INCLUDE '[.INC] TICOMN.INC/LIST'
C
C*********************************************************************
C
C       THIS SUBROUTINE FINISHES THE TWO-DIMENSIONAL PROGRAM BY WRITING
C       THE RESULTS AND THE POINTER SYSTEM IN ASCII FORM.
C
C*********************************************************************

C       SET THE PRINTOUT PARAMETER
        KPRINT = NINT(APASKY(19))

        IF (KPRINT .EQ. 2) THEN
           WRITE(JOUTAL,1000)
           WRITE(JOUTAL,1100) NITRE2, NNODG2, NCELG2, NBNDG2, NCELA2
           WRITE(JOUTAL,1200) TIMNTI
        ENDIF

        IF (KPRINT .GT. 2) CALL G2RESO

        CALL E2CORF

C       SAVE THE POINTER SYSTEM FOR THE FINAL TIME
        IF (KSRTE2 .LT. 1000) THEN
          WRITE(JTERMO,*) ' WRITTING ON FORMATTED POINTER FILE'
          CALL PSWRT2 (JPNTWR)
        ELSE
          WRITE(JTERMO,*) ' WRITTING ON UNFORMATTED POINTER FILE'
          CALL PSWRTU (JPNTWR)
        ENDIF
C
        WRITE(6,1300)  TIMNTI,TIMXTI
        WRITE(JOUTAL,1300)  TIMNTI,TIMXTI
        WRITE(JOUTAL,1400) EPS1TI,NGIVTI
```

```
      CALL TIMERR (JOUTAL, ZCUM, ' END OF RUN')
C
C     ----------------
C     FORMAT STATEMENTS
C     ----------------
C
1000  FORMAT('1'//)
1100  FORMAT(5X,'TOTAL NUMBER OF ITERATIONS     = ',I5,10X,
     1       5X,'TOTAL NUMBER OF NODES          = ',I5 /
     2       5X,'TOTAL NUMBER OF CELLS          = ',I5 ,10X,
     3       5X,'TOTAL NUMBER OF BOUNDARY NODES = ',I5 /
     4       5X,'TOTAL NUMBER OF CEWIC CELLS    = ',I5 /)
1200  FORMAT(5X,'TIME =',G14.5)
1300  FORMAT( ' TIME =',G14.5,5X,'TIMAX =',G14.5)
1400  FORMAT( ' EPS1TI =',G14.5,5X,'NGIVTI =',I5)

      STOP ' THE END'
      END
```

E2INI0

```
      SUBROUTINE E2INI0
C
      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] FRCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] H2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] KYCOMN.INC/LIST'
      INCLUDE '[.INC] TVCOMN.INC/LIST'
C
C**********************************************************************
C
C     THIS SUBROUTINE INITIALIZES ALL THE COMMON BLOCK ARRAYS THAT
C     ARE TO BE USED IN THE TWODO PROGRAM.  KSRTE2 INDICATES THE
C     RESTART PARAMETER WITH THE FOLLOWING MEANINGS
C        0 : A FRESH START USING G2INIT (ALGEBRAICALLY GENERATED GRID)
C        1 : A RESTART CASE
C        2 : FOR GENERATING C2HELP FILE
C        3 : A FRESH START USING G2IBLC (THE BLOCK GENERATED GRID)
C     THE ABOVE VALUES PLUS 1000 MEAN THAT THE INPUT/OUTPUT OF THE
C     DATA (PSREDU AND PSWRTU) IS DONE IN UNFORMATTED FORM, OTHERWISE
C     THE DATA (PSRED2 AND PSWRT2) IS IN FORMATTED FORM
C
C**********************************************************************
C
C     SETUP INPUT/OUTPUT UNITS
C
      CALL   SETUPU
```

```
C
C       SET UP INITIAL VALUES FOR REFERENCE TEMPERATURE TREFFL AND
C       PRESSURE PRESFL; SO THAT A RESTART CASE MAY BE ABLE TO CHANGE
C       THESE IF NEED BE
C
        APASKY(7) = 0.
        APASKY(9) = 0.
C
C       SET EITHER THE DEFAULT OPTIONS OR READ THEM
C
        CALL   E2OPTO
        CLOSE  (JREADI)

C       SET THE RETART PARAMETER
        KSRTE2 = IPASKY(6 )

C       SET THE PRINTOUT PARAMETER
        KPRINT = NINT(APASKY(19))

C       SET THE FUEL INJECTION PARAMETER
        IADDH2 = IPASKY(38)
C
C       SEE IF YOU WANT TO RESTART FROM A PREVIOUS RUN

        IF (KSRTE2 .EQ. 1 .OR. KSRTE2 .EQ. 1001) THEN
C
           IF (KSRTE2 .EQ. 1) THEN
               CALL PSRED2
           ELSE
               CALL PSREDU
           ENDIF
C
C          IF VARIABLE INLET BOUNDARY CONDITIONS ARE DESIRED THEN SET
C          THE VALUES FROM A PREVIOUS RUN
C
           IF (KPERFR .EQ. 1) CALL TVINI1
C
C          IF THE OPTION ROUTINES DID NOT SET THE FOLLOWING VALUES THEN
C          SET THESE ACCORDING TO THEIR PREVIOUS VALUES; OTHERWISE THEY
C          HAVE THE NEWLY RECOMMENDED VALUES WHICH CAN BE CHANGED BY THE
C          ROUTINE CHKREF
C
           IF (APASKY(7) .EQ. 0.) APASKY(7) = TREFFL
           IF (APASKY(9) .EQ. 0.) APASKY(9) = PRESFL
C
           CALL E2RSRT
           CALL E2CORI
C
C          TRANSPORT VALUES OF PHI AND RHOD WHICH WERE READ IN PSRED2
C
           APASKY(1) = CHNGE2(1,1)
           APASKY(2) = CHNGE2(1,2)
           GO TO 5
C
        ENDIF

C       SKIP THE INITIALIZATION PROCESS UNDER SPECIAL CONDITIONS, SUCH
```

```
C       AS WHEN GENERATING A C2HELP.DAT FILE

        IF (KSRTE2 .EQ. 2) RETURN
C
C       SETUP THE MAXIMUM AND MINIMUM VALUES OF ARTIFICIAL VISCOSITY
        SMAXE2 = APASKY( 1)
        SMINE2 = APASKY( 2)
        SDELE2 = 0.
C
C       SET THE EPSILON VALUE FOR CONVERGENCE CRITERION
        EPSLE2 = APASKY( 4)
C
C       SET THE RECIROCAL OF REYNOLDS NUMBER
        RREYE2 = APASKY(36)
C
C       SET THE RECIROCAL OF PRANDTL NUMBER
        RPRNE2 = APASKY(37)
C
C       SET THE RECIROCAL OF SCHMIDT NUMBER
        RSCHE2 = APASKY(38)

C       SET THE POWER FOR VISCOSITY POWER LAW
        OMEGE2 = APASKY(39)

C       SET THE GAMMA FACTOR FOR ENERGY EQUATION (G/(G-1))
        GFACE2 = APASKY(40)
C
C       SET THE PARAMETER FOR CONVERGENCE VARIABLE
C        1: AVERAGE   2: MAXIMUM   3: RMS
        KONVE2 = IPASKY(25)
C
C       SET THE PARAMETER FOR CONVERGENCE EQUATION VARIABLE
C        1: MASS  2: MOMENTUM   3: ENERGY  ETC.
        KEQNE2 = IPASKY(29)
C
C
C       SET THE MAXIMUM NUMBER OF ITERATIONS ALLOWED
        MITRE2 = IPASKY(5)

C       SET THE CURRENT NUMBER OF ITERATIONS
        NITRE2 = 0
C
C       SET THE PARAMETER INDICATING MAXIMUM NUMBER OF TEMPORAL CELL
C       LEVELS TO BE USED
        KHAFEZ = IPASKY(28)
C
C       INITIALIZE THE VALUES FOR THE CHEMISTRY ARRAYS
C
        CALL   C2INIT
C
C       INITIALIZE THE CHEMISTRY POINTER SYSTEM
C
        CALL   C2PONT
C
C       INITIALIZE THE VALUES FOR THE FLUID ARRAYS
C
        CALL   FLINI2
```

```
C
C       INITIALIZE THE LIGHT HILL MODEL VALUES IF NECESSARY
C
        CALL   LHINI2
C
C       INITIALIZE THE VALUES FOR THE GRID ARRAYS
C
        IF (KSRTE2 .EQ. 3 .OR. KSRTE2 .EQ. 1003) THEN
            CALL   G2IBLC
        ELSE
            CALL   G2INIT
        ENDIF

        IDBGE2 = IPASKY(14)
C
C       CHECK IF THE INITIAL GRID IS O.K.
C
        NERR   = 0
        CALL   CHKBN2 (0, 0, 0, 0, 0, NERR, '    ')
        CALL   CHKNN2 (0, 0, 0, 0, 0, NERR, '    ')
        CALL   CHKNC2 (0, 0, 0, 0, 0, NERR, '    ')
        IF (NERR .NE. 0) WRITE(6,*) ' WARNING: INITIAL GRID ERROR'
C
C       INITIALIZE THE VALUES FOR THE FREESTREAM VARIABLES
C
        CALL   FRINIT
C
C       INITIALIZE THE MAXIMUM ALLOWABLE SPECIES MASS-FRATIONS FOR
C       ALL SPECIES
C
        DO 4 IS = 1, MSPECH
            YMAXCH(IS) = 1.
4       CONTINUE
C
C       INITIALIZE THE VALUES FOR THE DEPENDENT VARIABLE ARRAYS
C
        CALL   DPINI2
C
C       INITIALIZE THE VALUES FOR THE ARRAYS USED IN SPATIAL
C       ADAPTATION PROCEDURE
C
        CALL   A2INIT
C
C       INITIALIZE THE VALUES FOR THE ARRAYS USED IN TEMPORAL
C       ADAPTATION PROCEDURE
C
        CALL   TIINI2
C
C       NCRSG2, FOR THE TIME ACCURATE PROBLEMS WILL INDICATE IF
C       CHARACTERISTIC BOUNDARY CONDITIONS ARE USED
C
C
5       DO 10 IN = 1, NBNDG2
           IF (IBNDG2(5,IN) .GE. 4 .AND. IBNDG2(5,IN) .LE. 7) THEN
C          IF (IBNDG2(5,IN) .GE. 3 .AND. IBNDG2(5,IN) .LE. 7) THEN
               NCRSG2 = 1
               GO TO 20
           ENDIF
```

```
10      CONTINUE
C
C       CALCULATE THE MAXIMUM ALLOWABLE SPECIES MASS-FRATIONS FOR
C       ALL SPECIES; FIRST INITIALIATION MAY HAVE TO BE DONE FOR THE
C       RESTART CASE
C
C
C       INITIALIZE THE MAXIMUM ALLOWABLE SPECIES MASS-FRATIONS FOR
C       ALL SPECIES
C
        DO 15 IS = 1, MSPECH
           YMAXCH(IS) = 1.
15      CONTINUE
C
        CALL CHKYMX
C
C       INITIALIZE THE CHANGE VARIABLES
C
20      DO 40 IQ = 1, MEQNFL
           DO 30 IN = 1, MNODG2
              CHNGE2(IQ,IN) = 0.
30         CONTINUE
40      CONTINUE
C       CALL A2CEWC
C
C       INITIALIZE THE ARTIFICIAL VISCOSITY AT EACH NODE

        DO 50 IN = 1, MNODG2
           SIGGE2(IN) = SMINE2
50      CONTINUE

        CALL E2DIFF
        CALL E2DIFF

C       INITIALIZE THE JACOBIAN METRICS, CELL VOLUMES (RECIPROCALS),
C       AND PERIMETERS FOR EACH CELL IN THE SPATIAL DOMAIN

        CALL M2AREA(0)

C       INITIALIZE THE RECIPROCALS OF MOLECULAR MASS FOR EACH SPECIES

        DO 60 IS = 1, NSPECH
           RAMWCH(IS) = 1./AMWTCH(IS)
60      CONTINUE
C
C       WRITE THE INITIAL PARAMETERS OF THE RUN
C
        IF (KPRINT .GT. 0) CALL WRINI2
        CALL ERINIT

        IF (KONVE2 .GT. 0) THEN
C            OPEN (UNIT=JHISTO, FILE='JHISTO.DAT', STATUS='NEW')
             WRITE (JHISTO,1500) MTITLE
        ENDIF

        CALL TIMERR (JOUTAL, ZCUM, 'INITIALIZATION COMPLETED')
C
```

```
C       PRINT OUT PARAMETERS
C          -
        IF (IDBGE2 .NE. 1 .AND. IDBGE2 .LT. 1000)  RETURN
        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) KSRTE2, KONVE2, MITRE2, NITRE2, KEQNE2,
     1                     KHAFE2, IDBGE2, IADDH2
        WRITE(JDEBUG,1400) SMAXE2, SMINE2, SDELE2, EPSLE2
        WRITE(JDEBUG,1500) MTITLE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM E2INIO' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'KSRTE2 = ',I5,15X,'KONVE2 = ',I5/
     1         5X,'MITRE2 = ',I5,15X,'NITRE2 = ',I5/
     2         5X,'KEQNE2 = ',I5,15X,'KHAFE2 = ',I5/
     3         5X,'IDBGE2 = ',I5,15X,'IADDH2 = ',I5/)
1400    FORMAT(5X,'SMAXE2 = ',G15.5,5X,'SMINE2 = ',G15.5/
     1         5X,'SDELE2 = ',G15.5,5X,'EPSLE2 = ',G15.5)
1500    FORMAT(A80)

        RETURN
        END
```

E2OPTO

```
        SUBROUTINE E2OPTO
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'KYCOMN.INC'
C
C*****************************************************************
C
C       THIS SUBROUTINE SETS THE DEFAULT OPTION PARAMETERS AND THEN
C       READS THE PARAMETERS TO BE CHANGED FROM GETKY2.
C
C*****************************************************************
C
C                  -------------------
C                  LIST  OF  PARAMETERS
C                  -------------------
C       THE FOLLOWING PARAMETERS COULD BE SET BY SUBROUTINE GETKY2
C
C                      KYWRDA   DEFINED BY APASKY
C               1.   'SMAXE2='      ,    2.    'SMINE2='      ,
C               3.   'CFLNTI='      ,    4.    'EPSLE2='      ,
```

```
C              5.   'AMCHFL='     ,    6.    'RHORFL='     ,
C              7.   'TREFFL='     ,    8.    'TREFCH='     ,
C              9.   'PRESFL='     ,    10.   'PRESCH='     ,
C              11.  'DISTFL='     ,    12.   'TEMP1C='     ,
C              13.  'TEMP2C='     ,    14.   'TEMP3C='     ,
C              15.  'ALPHA2='     ,    16.   'BETAA2='     ,
C              17.  'GAMMA2='     ,    18.   'DELTA2='     ,
C              19.  'PRINTO='     ,    20.   'TIMXTI='     ,
C              21.  'PBPIFR='     ,    22.   'EPS1TI='     ,
C              23.  'EPSOTI='     ,    24.   'TIMNTI='     ,
C              25.  'TRIGCH='     ,    26.   'ERRMIN='     ,
C              27.  'ERRMTI='     ,    28.   'EPS1MN='     ,
C              29.  'EPS1MX='     ,    30.   'RHORFR='     ,
C              31.  'UCOMFR='     ,    32.   'VCOMFR='     ,
C              33.  'PRESFR='     ,    34.   'FCTRTI='     ,
C              35.  'DTCNTI='     ,    36.   'RREYE2='     ,
C              37.  'RPRNE2='     ,    38.   'RSCHE2='     ,
C              39.  'OMEGE2='     ,    40.   'GFACE2='     ,
C              41.  'CFLXTI='     ,    42.   '      ='     ,
C
C
C                     KYWRDI   DEFINED BY IPASKY
C              1.   'NREACH='     ,    2.    'NSPECH='     ,
C              3.   'KROGER='     ,    4.    'KORDER='     ,
C              5.   'MITRE2='     ,    6.    'KSRTE2='     ,
C              7.   'NGIVTI='     ,    8.    'METHA2='     ,
C              9.   'JREADS='     ,    10.   'KTIMTI='     ,
C              11.  'K1ADA2='     ,    12.   'K2ADA2='     ,
C              13.  'KDEBUG='     ,    14.   'IDBGE2='     ,
C              15.  'IDBGA2='     ,    16.   'MTHRA2='     ,
C              17.  'KFACTI='     ,    18.   'NINRCH='     ,
C              19.  'KDPENI='     ,    20.   'KPLTA2='     ,
C              21.  'IDBGFR='     ,    22.   'NXTDA2='     ,
C              23.  'MALVG2='     ,    24.   'KADPTI='     ,
C              25.  'KONVE2='     ,    26.   'MITRA2='     ,
C              27.  'MITRPS='     ,    28.   'KHAFEZ='     ,
C              29.  'KEQNE2='     ,    30.   'KMERA2='     ,
C              31.  'KCHKA2='     ,    32.   'MITEPS='     ,
C              33.  'IMPLTI='     ,    34.   'IDBGTI='     ,
C              35.  'KPERFR='     ,    36.   'MCYCFR='     ,
C              37.  'IDBGG2='     ,    38.   'IADDH2='     ,
C              39.  'KDIFTI='     ,    40.   'KBLOCK='     ,
C              41.  '      ='     ,    42.   '      ='     ,
C
C       ---------------------------------
C       DEFAULT  VALUES  :  REAL VARIABLES
C       ---------------------------------
C
C       SET THE DEFAULT VALUES FIRST : REAL VARIABLES
C
C1      MAXIMUM COEFFICIENT OF ARTIFICIAL VISCOSITY
        SMAXE2 = 0.1
C
C2      MINIMUM COEFFICIENT OF ARTIFICIAL VISCOSITY
        SMINE2 = 0.05
C
C3      MINIMUM CFL NUMBER CFL
```

```
      CFLNTI = 0.9
C          -
C4    MINIMUM CRITERIA FOR CONVERGENCE
      EPSLE2 = 1.E-4
C
C5    REFERENCE MACH NUMBER
      AMCHFL = 1.
C
C6    REFERENCE FLUID DENSITY
      RHORFL = 1.
C
C7    REFERENCE FLUID TEMPERATURE
      TREFFL = 1.
C
C8    REFERENCE CHEMISTRY TEMPERATURE
      TREFCH = 298.
C
C9    REFERENCE FLUID PRESSURE
      PRESFL = 1.
C
C10   REFERENCE CHEMISTRY PRESSURE
      PRESCH = 1.E05
C
C11   REFERENCE FLUID CHARACTERISTIC LENGTH
      DISTFL = 1.
C
C12   TEMPERATURE USED TO DETERMINE BACKWARD (OR FORWARD) RATES
      TEMP1C =  298.
      TEMP2C = 2000.
      TEMP3C = 3000.
C
C15   CONSTANTS FOR ADAPTATION
      ALPHA2 = 1.
      BETAA2 = 1.25
      GAMMA2 = 0.75
      DELTA2 = 0.25

C19   SEE IF YOU WANT TO WRITE EXTRA OUTPUT (G2RESO AND WRINI2)
      PRINTO = 2.

C20   MAXIMUM TIME OF THE RUN
      TIMXTI = 1.

C21   BACK PRESSURE RATIO (PBd/PREFFL OR PB/PRESFR)
      PBPIFR = 0.

C22   EPSILON USED FOR TEMPORAL RESOLUTION
      EPS1TI = 1.

C23   EPSILON CORRECTION FOR ZERO VALUE OF TEMPORAL CRITERION
      EPSOTI = 0.1

C24   MINIMUM TIME OF THE RUN
      TIMNTI = 0.

C25   TRIGGER TEMPERATURE FOR CHEMISTRY (FROZEN BELOW TRIGCH)
      TRIGCH = 0.
```

```
C26     MINIMUM ERROR BELOW WHICH EPS1TI WILL BE INCREASED
        ERRMIN = 1.E-6

C27     MAXIMUM ERROR ABOVE WHICH EPS1TI WILL BE DECREASED
        ERRMTI = 0.1

C28     MINIMUM ALLOWABLE VALUE OF EPS1TI
        EPS1MN = 0.5

C29     MAXIMUM ALLOWABLE VALUE OF EPS1TI
        EPS1MX = 0.00001

C30     FREE STREAM DENSITY (NON-DIMENSIONAL)
        RHORFR = 1.

C31     FREE STREAM VELOCITY (NON-DIMENSIONAL X COMPONENT)
        UCOMFR = 0.

C32     FREE STREAM VELOCITY (NON-DIMENSIONAL Y COMPONENT)
        VCOMFR = 0.

C33     FREE STREAM PRESSURE (NON-DIMENSIONAL)
        PRESFR = 1.

C34     FACTOR FOR ADJUSTING THE CELL TIME STEPS
        FCTRTI = 1.

C35     SET THE CONSTANT CELL TIME STEP; NEGATIVE VALUE MEANS THAT
C       A LOCAL VALUE WILL BE COMPUTED
        DTCNTI = -1.

C36     SET THE RECIROCAL OF REYNOLDS NUMBER
        RREYE2 = 0.

C37     SET THE RECIROCAL OF PRANDTL NUMBER
        RPRNE2 = 1.

C38     SET THE RECIROCAL OF SCHMIDT NUMBER
        RSCHE2 = 1.

C39     SET THE POWER FOR VISCOSITY POWER LAW
        OMEGE2 = 1.

C40     SET THE GAMMA FACTOR FOR ENERGY EQUATION (G/(G-1))
        GFACE2 = 3.5
C
C41     MAXIIMUM CFL NUMBER CFL
        CFLXTI = 2.0


C***    SET UP THE ABOVE CONSTANTS IN THE PASS VARIABLE ***

        APASKY( 1) = SMAXE2
        APASKY( 2) = SMINE2
        APASKY( 3) = CFLNTI
        APASKY( 4) = EPSLE2
```

```
      APASKY( 5) = AMCHFL
      APASKY( 6) = RHORFL
      APASKY( 7) = TREFFL
      APASKY( 8) = TREFCH
      APASKY( 9) = PRESFL
      APASKY(10) = PRESCH
      APASKY(11) = DISTFL
      APASKY(12) = TEMP1C
      APASKY(13) = TEMP2C
      APASKY(14) = TEMP3C
      APASKY(15) = ALPHA2
      APASKY(16) = BETAA2
      APASKY(17) = GAMMA2
      APASKY(18) = DELTA2
      APASKY(19) = PRINTO
      APASKY(20) = TIMXTI
      APASKY(21) = PBPIFR
      APASKY(22) = EPS1TI
      APASKY(23) = EPSOTI
      APASKY(24) = TIMNTI
      APASKY(25) = TRIGCH
      APASKY(26) = ERRMIN
      APASKY(27) = ERRMTI
      APASKY(28) = EPS1MN
      APASKY(29) = EPS1MX
      APASKY(30) = RHORFR
      APASKY(31) = UCOMFR
      APASKY(32) = VCOMFR
      APASKY(33) = PRESFR
      APASKY(34) = FCTRTI
      APASKY(35) = DTCNTI
      APASKY(36) = RREYE2
      APASKY(37) = RPRNE2
      APASKY(38) = RSCHE2
      APASKY(39) = OMEGE2
      APASKY(40) = GFACE2
      APASKY(41) = CFLXTI

C     ---------------------------------------
C     DEFAULT  VALUES  :  INTEGRAL  VARIABLES
C     ---------------------------------------
C
C     SET THE DEFAULT VALUES : INTEGER VARIABLES
C
C1    NUMBER OF REACTIONS
      NREACH = 0
C
C2    NUMBER OF SPECIES (INCLUDING INERT SPECIES)
      NSPECH = 0
C
C3    PARAMETER INDICATING THE TYPE OF CHEMISTRY MODEL TO BE USED
      KROGER = 0
C
C4    PARAMETER INDICATING IF THERE ARE NON-ELEMENTARY REACTIONS
      KORDER = 0
C
```

```
C5      MAXIMUM NUMBER OF ITERATIONS
        MITRE2 = 10000
C
C6      PARAMETER INDICATING IF THE FLOW IS TO RE-STARTED
        KSRTE2 = 0
C
C7      MAXIMUM GIVEN LEVEL FOR TEMPORAL EMBEDDING
        NGIVTI = 0
C
C8      VARIATION METHA2 FOR ADAPTATION
C         (VALUE, GRADIENT, LAPLACIAN, MARSHA BURGER -- EVEN FOR CELLS)
        METHA2 = 4
C
C9      UNIT FOR READING THE SCHEDULE INPUT PROGRAM
        JREADS = 0
C
C10     PARAMETER INDICATING IF RESULTS AT VARIOUS TIME INTERVALS
C       ARE NEEDED
        KTIMTI = 0
C
C11     KEY VARIABLE FOR SPATIAL ADAPTATION -- DPENG2(1,I)
        K1ADA2 = 1
C
C12     KEY VARIABLE FOR SPATIAL ADAPTATION -- DPENG2(1,I)
        K2ADA2 = 0
C
C13     OUTPUT (DEBUG) PARAMETER
        KDEBUG = 0
C
C14     DEBUG PARAMETER FOR EULER ROUTINES (E2 ROUTINES)
        IDBGE2 = 0
C
C15     DEBUG PARAMETER FOR ADAPTIVE ROUTINES (A2 ROUTINES)
        IDBGA2 = 0
C
C16     THE NUMBER OF TIMES OF ADAPTATION CYCLES AFTER WHICH THE
C       THRESHOLD LIMITS WILL BE COMPUTED
        MTHRA2 = 1
C
C17     PARAMETER INDICATING IF THE CELL TIME STEPS ARE TO RE-ADJUSTED
        KFACTI = 0
C
C18     NUMBER OF INERT SPECIES
        NINRCH = 0
C
C19     OPTION PARAMETER FOR SETTING DEPENDENT VARIABLES
C           1: READ FROM INPUT FILE -- AT ALL NODES
C           2: SET UNIFORM VALUES
C           3: SET LINEARLY VARYING VALUES FROM INLET TO OUTLET
        KDPENI = 1
C
C20     PARAMETER INDICATING IF THRESHOLD PLOTS ARE NEEDED
        KPLTA2 = 0
C
C21     DEBUG PARAMETER FOR FREE STREAM CONDITIONS
        IDBGFR = 0
C
```

```
C22     NUMBER OF CELLS TO BE EXTENDED FOR ADAPTIVE GRIDS
        NXTDA2 = 0
C
C23     NUMBER OF MAXIMUM FINE LEVELS TO BE USED FOR ADAPTIVE GRIDS
        MALVG2 = 3
C
C24     KEY VARIABLE FOR TEMPORAL ADAPTATION -- BIGWG2(4,I)
        KADPTI = 4
C
C25     CONVERGENCE CRITERIA TYPE VARIABLE
C        1: AVERAGE     2: MAXIMUM      3: RMS
C       DEFAULT IS ZERO (NONE) FOR TIME-ACCURATE PROBLEMS
        KONVE2 = 0
C
C26     THE MAXIMUM NUMBER OF TIMES BEFORE SPATIAL ADAPTATION IS DONE
        MITRA2 = 100
C
C27     THE MAXIMUM NUMBER OF TIMES BEFORE THE POINTER SYSTEM IS SAVED
        MITRPS = 100
C
C28     OPTION PARAMETER FOR HAFEZ DOMINANT EIGENVALUE
        KHAFEZ = 0
C
C29     PARAMETER DENOTING THE EQUATION FOR WHICH CONVERGENCE HISTORY
C       IS WRITTEN BY ROUTINE E2CONO, DEFAULT IS MOMEMTUM EQUATION
        KEQNE2 = 2
C
C30     PARAMETER INDICATING IF THE COLLAPSING OF CELLS IS TO BE DONE
        KMERA2 = 1
C
C31     DEBUG PARAMETER FOR CHECKING THE SUPERCELL AND NEIGHBOUR-
C       CELL CALCULATIONS.  INPUT IN BINARY CODED VALUE
C        1: CHECK SUPERCELL  2: CHECK NEIGHBOUR-CELL
C        4: CHECK BEFORE COLLAPSE
        KCHKA2 = 0

C32     MAXIMUM NUMBER OF ITERATIONS AFTER WHICH EPS1TI IS DECREASED
        MITEPS = 10

C33     PARAMETER INDICATING IF IMPLICIT SOURCE TERMS ARE TO BE USED
C       IMPLTI: 1 FOR EXPLICIT; 0 FOR IMPLICIT
        IMPLTI = 1

C34     DEBUG PARAMETER FOR TEMPORAL ROUTINES (TI ROUTINES)
        IDBGTI = 0

C35     PARAMETER INDICATING IF PERIODIC BOUNDARY CONDITIONS ARE TO BE
C       USED
        KPERFR = 0

C36     MAXIMUM NUMBER OF CYCCLES FOR PERIODIC BOUNDARY CONDITIONS
        MCYCFR = 20000

C
C37     DEBUG PARAMETER FOR GRID ROUTINES (G2 ROUTINES)
        IDBGG2 = 0
```

```
C
C38     FUEL INJECTION PARAMETER
C          0 : NO FUEL
C          1 : FUEL INJECTED FOR FIRST TIME
C          2 : FUEL INJECTED AFTER FIRST TIME (RESTART CASE)
        IADDH2 = 0

C39     PARAMETER INDICATING IF A DIFFERNCE OF MASS FRACTION CRITERIA
C        (SHEAR LAYER CRITERIA) IS TO BE USED FOR LIMITING CELL TIME-STEPS
        KDIFTI = 0

C40     PARAMETER INDICATING IF A SMALL BLOCK OF CELLS IS TO BE INTEGRATED
C       AT ONE TIME, INSTEAD OF THE WHOLE DOMAIN.  THIS IS USEFUL IN
C       ACCELERATING THE CONVERGENCE TO STEADY STATE AND FOR PROBELMS
C       WHICH ARE PREDOMINANTLY "PARABOLIC" IN NATURE
        KBLOCK = 0

C***    SET UP THE ABOVE CONSTANTS IN THE PASS VARIABLE ***

        IPASKY( 1) = NREACH
        IPASKY( 2) = NSPECH
        IPASKY( 3) = KROGER
        IPASKY( 4) = KORDER
        IPASKY( 5) = MITRE2
        IPASKY( 6) = KSRTE2
        IPASKY( 7) = NGIVTI
        IPASKY( 8) = METHA2
        IPASKY( 9) = JREADS
        IPASKY(10) = KTIMTI
        IPASKY(11) = K1ADA2
        IPASKY(12) = K2ADA2
        IPASKY(13) = KDEBUG
        IPASKY(14) = IDBGE2
        IPASKY(15) = IDBGA2
        IPASKY(16) = MTHRA2
        IPASKY(17) = KFACTI
        IPASKY(18) = NINRCH
        IPASKY(19) = KDPENI
        IPASKY(20) = KPLTA2
        IPASKY(21) = IDBGFR
        IPASKY(22) = NXTDA2
        IPASKY(23) = MALVG2
        IPASKY(24) = KADPTI
        IPASKY(25) = KONVE2
        IPASKY(26) = MITRA2
        IPASKY(27) = MITRPS
        IPASKY(28) = KHAFEZ
        IPASKY(29) = KEQNE2
        IPASKY(30) = KMERA2
        IPASKY(31) = KCHKA2
        IPASKY(32) = MITEPS
        IPASKY(33) = IMPLTI
        IPASKY(34) = IDBGTI
        IPASKY(35) = KPERFR
        IPASKY(36) = MCYCFR
        IPASKY(37) = IDBGG2
        IPASKY(38) = IADDH2
```

```
      IPASKY(39) = KDIFTI
      IPASKY(40) = KBLOCK

C
C     ----------
C     ECHO  PRINT
C     ----------
C
C     ECHO PRINT THE INPUT PARAMETERS -- ALL COMMENTS MUST HAVE AN
C     ASTERISK IN THE FIRST COLUMN

      CALL IMAGEI (JOUTAL,JREADI,MTITLE)

C     GET THE CHANGED VARIABLES

      CALL GETKY2
C
C     KEEP THE FOLLOWING VARIABLES ALWAYS IN INPUTI.DAT FOR
C     INDENTIFICATION
      NREACH     = IPASKY( 1)
      NSPECH     = IPASKY( 2)
      KROGER     = IPASKY( 3)
      JREADS     = IPASKY( 9)
      NINRCH     = IPASKY(18)
C
C     PRINT OUT PARAMETERS
C
      KDEBUG = IPASKY(13)
      IF (KDEBUG .NE. 2 .AND. KDEBUG .LT. 1000)  RETURN

      WRITE(JDEBUG,1000)
      WRITE(JDEBUG,1100)
      WRITE(JDEBUG,1200)
      WRITE(JDEBUG,1300)
      WRITE(JDEBUG,1400) (APASKY(K),K=1,40)
      WRITE(JDEBUG,1500)
      WRITE(JDEBUG,1600) (IPASKY(K),K=1,40)

C     -----------------
C     FORMAT STATEMENTS
C     -----------------

1000  FORMAT(//10X,'----------------------' )
1100  FORMAT(  10X,'DEBUG PRINT FROM E20PTO' )
1200  FORMAT(  10X,'----------------------'/)
1300  FORMAT(  10X,'APASKY ARRAY'/)
1400  FORMAT(  8E15.6)
1500  FORMAT(  10X,'IPASKY ARRAY'/)
1600  FORMAT(  16I5)

C     ALTERNATE VALUES FOR IPASKY AND APASKY
C     IPASKY(12) : IDBGA2

      RETURN
      END
```

E2PRMU

```
      SUBROUTINE E2PRMT (INODE, ITYPE)
C                E2PRMU

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'PRCOMN.INC'
      INCLUDE 'IOCOMN.INC'

C*********************************************************************

C     THIS SUBROUTINE COMPUTES THE PRIMITIVE VARIABLES AT A GIVEN
C     NODE 'INODE'.  THE VARIABLE 'ITYPE' DETERMINES THE TYPE OF
C     CALCULATIONS THAT MIGHT BE NEEDED.

C*********************************************************************

      RHORPR = DPENG2(1,INODE)
      UCOMPR = DPENG2(2,INODE)/RHORPR
      VCOMPR = DPENG2(3,INODE)/RHORPR
      BEPSPR = DPENG2(4,INODE)
      BE     = BEPSPR/RHORPR
      VELO2U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BE
      VELO2  = FMREFL*VELO2U

C     COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

      SUMY   = 0.

      DO 10 IS = 1, NEQSCH

         JS         = NEQBAS + IS
         YSPEPR(IS) = DPENG2(JS,INODE)/DPENG2(1,INODE)

         IF (YSPEPR(IS) .LT. 0.) THEN
           YSPEPR(IS)       = 0.
           DPENG2(JS,INODE) = 0.
         ENDIF

         IF (YSPEPR(IS) .GT. YMAXCH(IS)) THEN
           YSPEPR(IS)       = YMAXCH(IS)
           DPENG2(JS,INODE) = YMAXCH(IS)*DPENG2(1,INODE)
         ENDIF

         SUMY       = SUMY + YSPEPR(IS)
10    CONTINUE
```

```
      YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
C     YSPEPR(NEQSCH+1) = ABS(1. - SUMY - YNRTCH)
      IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
      IF (YSPEPR(NEQSCH+1) .GT. YMAXCH(NEQSCH+1))
     1       YSPEPR(NEQSCH+1) = YMAXCH(NEQSCH+1)

      SYSHFS = 0.
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
C     COMPUTE THE TEMPERATURE IN DEGREE K
C
      DO 20 IS = 1, NSPECH
          SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
          SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
          SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
          BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
20    CONTINUE

      BIGBM   = SYSCPS - UGASFL*SYSBMS
      BIGCM   = BE - 0.5*VEL02 - SYSHFS + TREFCH*SYSCPS
     1                + 0.5*TREFCH*TREFCH*BIGAM
      IF (BIGAM .LT. 1.E-10) THEN
         TEMP   = BIGCM/BIGBM
      ELSE
         DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
         TEMP   = ( SQRT(DISCRI)-BIGBM )/BIGAM
      ENDIF
C
C     NORMALIZE THE TEMPERATURE
C
      TEMPPR = TEMP/TREFFL
C
C     COMPUTE THE DIMENSIONLESS PRESSURE
C
      PRESPR  = RHORPR*TEMPPR*AMWTFL*SYSBMS
C     IF (PRESPR .LE. 0.) CALL CHKPR2(INODE)
C
C     SAVE THE PRESSURE AND TEMPERATURE AT THE NODE
C
      PRESG2(INODE) = PRESPR
      TEMPG2(INODE) = TEMPPR

      GOTO (40,30,30),ITYPE
C
30    BIGAMT = BIGAM*TEMP
      SYSCVS = BIGBM + BIGAMT
      GAMAPR = (SYSCPS+BIGAMT)/SYSCVS
      SONDPR = GAMAPR*PRESPR/RHORPR
      SONDPR = SQRT(SONDPR)

      IF (ITYPE .EQ. 2) RETURN
      AMCHPR = SQRT(VEL02U)/SONDPR

40    RETURN
      END
```

E2PRMT

```
      SUBROUTINE E2PRMT (INODE, ITYPE)

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] PRCOMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      DATA KOUNT /0/

C*********************************************************************

C     THIS SUBROUTINE COMPUTES THE PRIMITIVE VARIABLES AT A GIVEN
C     NODE 'INODE'.  THE VARIABLE 'ITYPE' DTERMINES THE TYPE OF
C     CALCULATIONS THAT MIGHT BE NEEDED.

C*********************************************************************
C
      RHORPR = DPENG2(1,INODE)
      UCOMPR = DPENG2(2,INODE)/RHORPR
      VCOMPR = DPENG2(3,INODE)/RHORPR
      BEPSPR = DPENG2(4,INODE)
      BE     = BEPSPR/RHORPR
      VEL02U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BE
      VEL02  = FMREFL*VEL02U

C     COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

      SUMY   = 0.
      YUPPER = 1. - YNRTCH

      DO 10 IS = 1, NEQSCH

          JS         = NEQBAS + IS
          YSPEPR(IS) = DPENG2(JS,INODE)/DPENG2(1,INODE)

          IF (YSPEPR(IS) .LT. 0.) THEN
            YSPEPR(IS)       = 0.
            DPENG2(JS,INODE) = 0.
          ENDIF

          IF (YSPEPR(IS) .GT. YUPPER) THEN
            YSPEPR(IS)       = YUPPER
            DPENG2(JS,INODE) = YUPPER*DPENG2(1,INODE)
```

```
          ENDIF
          -
          SUMY       = SUMY + YSPEPR(IS)

10      CONTINUE

        YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
C       YSPEPR(NEQSCH+1) = ABS(1. - SUMY - YNRTCH)
        IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.

        SYSHFS = 0.
        SYSCPS = 0.
        SYSBMS = 0.
        BIGAM  = 0.
C
C       COMPUTE THE TEMPERATURE IN DEGREE K
C
        DO 20 IS = 1, NSPECH
            SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)/AMWTCH(IS)
            SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
            SYSBMS = SYSBMS + YSPEPR(IS)/AMWTCH(IS)
            BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
20      CONTINUE

        BIGBM   = SYSCPS - UGASFL*SYSBMS
        BIGCM   = BE - 0.5*VEL02 - SYSHFS + TREFCH*SYSCPS
     1                 + 0.5*TREFCH*TREFCH*BIGAM

        IF (BIGAM .LT. 1.E-10) THEN
           TEMP   = BIGCM/BIGBM
        ELSE
           DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
           TEMP   = ( SQRT(DISCRI)-BIGBM )/BIGAM
        ENDIF
C
C       NORMALIZE THE TEMPERATURE
C
        TEMPPR = TEMP/TREFFL
C
C       COMPUTE THE DIMENSIONLESS PRESSURE
C
        PRESPR  = RHORPR*TEMPPR*AMWTFL*SYSBMS
C
        IF (PRESPR .LE. 0.) CALL CHKPR2(INODE)
C
C       SAVE THE PRESSURE AND TEMPERATURE AT THE NODE
C
        PRESG2(INODE) = PRESPR
        TEMPG2(INODE) = TEMPPR

        GOTO (40,30,30),ITYPE
C
30      SYSCVS = BIGBM + BIGAM *TEMP
        GAMAPR = SYSCPS/SYSCVS
        SONDPR = GAMAPR*PRESPR/RHORPR

        IF(SONDPR .LT. 0.) THEN
```

```
          ZER1 = SONDPR
          ZER2 = TEMPPR
          WRITE(JDEBUG,1000) RHORPR,BEPSPR,UCOMPR,VCOMPR,TEMPPR,PRESPR,
     1      GAMAPR,SONDPR,GEOMG2(1,INODE),GEOMG2(2,INODE),INODE

          CALL ERRORM(3,'E2PRMT','SOUND2',ZER1,'TEMP  ',ZER2,JPRINT,
     1      'SPEED OF SOUND IS NEGATIVE')

        ENDIF

        SONDPR = SQRT(SONDPR)

        IF (ITYPE .EQ. 2) GOTO 40

        AMCHPR = SQRT(VEL02U)/SONDPR
C
C       PRINT OUT PARAMETERS
C

40      IF (IDBGE2 .NE. 7 .AND. IDBGE2 .LT. 1000)  RETURN

        IF (KOUNT .EQ. 0) THEN
           KOUNT = 1
           WRITE(JDEBUG,1100)
           WRITE(JDEBUG,1200)
           WRITE(JDEBUG,1300)
        ENDIF

        WRITE(JDEBUG,1400) INODE,  RHORPR, BEPSPR, UCOMPR, VCOMPR,
     1                             TEMPPR, PRESPR, YNRTCH

        IF (ITYPE .NE. 1) THEN
          WRITE(JDEBUG,1500) GAMAPR, SONDPR, AMCHPR
        ENDIF

C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(' RHORPR =',E15.6,10X,' BEPSPR =',E15.6/
     1         ' UCOMPR =',E15.6,10X,' VCOMPR =',E15.6/
     1         ' TEMPPR =',E15.6,10X,' PRESPR =',E15.6/
     1         ' GAMAPR =',E15.6,10X,' SOUND2 =',E15.6/
     1         ' XDIS   =',E15.6,10X,' YDIS   =',E15.6/
     5         ' NODE   =',I10)
1100    FORMAT(//10X,'----------------------' )
1200    FORMAT(  10X,'DEBUG PRINT FROM E2PRMT' )
1300    FORMAT(  10X,'----------------------'/)
1400    FORMAT(5X, 'NODE  =',   I5, 10X, 'RHORPR=', G14.5,
     1         5X, 'BEPSPR=', G14.5, 5X, 'UCOMPR=', G14.5/
     1         5X, 'VCOMPR=', G14.5, 5X, 'TEMPPR=', G14.5/
     2         5X, 'PRESPR=', G14.5, 5X, 'YNRTPR=', G14.5/ )
1500    FORMAT(5X, 'GAMAPR=', G14.5, 5X, 'SONDPR=', G14.5,
     1         5X, 'AMCHPR=', G14.5                        )

        RETURN
        END
```

E2RSRT

```
      SUBROUTINE E2RSRT
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'H2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
      INCLUDE 'TICOMN.INC'

      CHARACTER*7  KYWRDA(NAPAKY)      ,   KYWRDI(NIPAKY)
      DATA KYWRDA/ 'SMAXE2=', 'SMINE2=',  'CFLNTI=', 'EPSLE2=',
     1             'AMCHFL=', 'RHORFL=',  'TREFFL=', 'TREFCH=',
     2             'PRESFL=', 'PRESCH=',  'DISTFL=', 'TEMP1C=',
     3             'TEMP2C=', 'TEMP3C=',  'ALPHA2=', 'BETAA2=',
     4             'GAMMA2=', 'DELTA2=',  'PRINTO=', 'TIMXTI=',
     5             'PBPIFR=', 'EPS1TI=',  'EPSOTI=', 'TIMNTI=',
     6             'TRIGCH=', 'ERRMIN=',  'ERRMTI=', 'EPS1MN=',
     7             'EPS1MX=', 'RHORFR=',  'UCOMFR=', 'VCOMFR=',
     8             'PRESFR=', 'FCTRTI=',  'DTCNTI=', 'RREYE2=',
     9             'RPRNE2=', 'RSCHE2=',  'OMEGE2=', 'GFACE2=',
     *             'CFLXTI=', '      ='/

      DATA KYWRDI/ 'NREACH=', 'NSPECH=',  'KROGER=', 'KORDER=',
     1             'MITRE2=', 'KSRTE2=',  'NGIVTI=', 'METHA2=',
     2             'JREADS=', 'KTIMTI=',  'K1ADA2=', 'K2ADA2=',
     3             'KDEBUG=', 'IDBGE2=',  'IDBGA2=', 'MTHRA2=',
     4             'KFACTI=', 'NINRCH=',  'KDPENI=', 'KPLTA2=',
     5             'IDBGFR=', 'NXTDA2=',  'MALVG2=', 'KADPTI=',
     6             'KONVE2=', 'MITRA2=',  'MITRPS=', 'KHAFEZ=',
     7             'KEQNE2=', 'KMERA2=',  'KCHKA2=', 'MITEPS=',
     8             'IMPLTI=', 'IDBGTI=',  'KPERFR=', 'MCYCFR=',
     9             'IDBGG2=', 'IADDH2=',  'KDIFTI=', 'KBLOCK=',
     *             '      =', '      =',/
C
C*********************************************************************
C
C       THIS SUBROUTINE INITIALIZES ALL THE OPTION PARAMETERS FOR THE
C       RESTART CASE,  THESE PARAMETERS ARE THE ONES WHICH YOU MIGHT
C       WANT TO CHANGE SECOND TIME AROUND.  IF A PARAMETER IS DIFFERENT
C       IN THE EARLIER AND CURRENT RUN THEN IT MUST BE DEFINED BOTH THE
C       TIMES IN THE INPUT FILE INPUTI.DAT; UNLESS IF THE PARAMETER
C       HAPPENS TO BE THE DEFAULT VALUE.
C
```

```
C*********************************************************************
C
      IF ( MARIKY( 5) .NE. 0 )  MITRE2 = IPASKY( 5)
      IF ( MARIKY( 6) .NE. 0 )  KSRTE2 = IPASKY( 6)
      IF ( MARIKY( 7) .NE. 0 )  NGIVTI = IPASKY( 7)
      IF ( MARIKY( 8) .NE. 0 )  METHA2 = IPASKY( 8)
      IF ( MARIKY( 9) .NE. 0 )  JREADS = IPASKY( 9)
      IF ( MARIKY(10) .NE. 0 )  KTIMTI = IPASKY(10)
      IF ( MARIKY(11) .NE. 0 )  K1ADA2 = IPASKY(11)
      IF ( MARIKY(12) .NE. 0 )  K2ADA2 = IPASKY(12)
      IF ( MARIKY(13) .NE. 0 )  IDBGFL = IPASKY(13)
      IF ( MARIKY(14) .NE. 0 )  IDBGE2 = IPASKY(14)
      IF ( MARIKY(15) .NE. 0 )  IDBGA2 = IPASKY(15)
      IF ( MARIKY(16) .NE. 0 )  MTHRA2 = IPASKY(16)
      IF ( MARIKY(17) .NE. 0 )  KFACTI = IPASKY(17)
      IF ( MARIKY(20) .NE. 0 )  KPLTA2 = IPASKY(20)
      IF ( MARIKY(21) .NE. 0 )  IDBGFR = IPASKY(21)
      IF ( MARIKY(22) .NE. 0 )  NXTDA2 = IPASKY(22)
      IF ( MARIKY(23) .NE. 0 )  MALVG2 = IPASKY(23)
      IF ( MARIKY(24) .NE. 0 )  KADPTI = IPASKY(24)
      IF ( MARIKY(25) .NE. 0 )  KONVE2 = IPASKY(25)
      IF ( MARIKY(26) .NE. 0 )  MITRA2 = IPASKY(26)
      IF ( MARIKY(27) .NE. 0 )  MITRPS = IPASKY(27)
      IF ( MARIKY(28) .NE. 0 )  KHAFEZ = IPASKY(28)
      IF ( MARIKY(29) .NE. 0 )  KEQNE2 = IPASKY(29)
      IF ( MARIKY(30) .NE. 0 )  KMERA2 = IPASKY(30)
      IF ( MARIKY(31) .NE. 0 )  KCHKA2 = IPASKY(31)
      IF ( MARIKY(33) .NE. 0 )  IMPLTI = IPASKY(33)
      IF ( MARIKY(34) .NE. 0 )  IDBGTI = IPASKY(34)
C     IF ( MARIKY(35) .NE. 0 )  KPERFR = IPASKY(35)
      IF ( MARIKY(36) .NE. 0 )  MCYCFR = IPASKY(36)
      IF ( MARIKY(37) .NE. 0 )  IDBGG2 = IPASKY(37)
      IF ( MARIKY(38) .NE. 0 )  IADDH2 = IPASKY(38)
      IF ( MARIKY(39) .NE. 0 )  KDIFTI = IPASKY(39)
      IF ( MARIKY(40) .NE. 0 )  KBLOCK = IPASKY(39)

      IF ( MARAKY( 1) .NE. 0 )  SMAXE2 = APASKY( 1)
      IF ( MARAKY( 2) .NE. 0 )  SMINE2 = APASKY( 2)
      IF ( MARAKY( 3) .NE. 0 )  CFLNTI = APASKY( 3)
      IF ( MARAKY( 4) .NE. 0 )  EPSLE2 = APASKY( 4)
      IF ( MARAKY(15) .NE. 0 )  ALPHA2 = APASKY(15)
      IF ( MARAKY(16) .NE. 0 )  BETAA2 = APASKY(16)
      IF ( MARAKY(17) .NE. 0 )  GAMMA2 = APASKY(17)
      IF ( MARAKY(18) .NE. 0 )  DELTA2 = APASKY(18)
      IF ( MARAKY(20) .NE. 0 )  TIMXTI = APASKY(20)
      IF ( MARAKY(22) .NE. 0 )  EPS1TI = APASKY(22)
      IF ( MARAKY(23) .NE. 0 )  EPSOTI = APASKY(23)
      IF ( MARAKY(25) .NE. 0 )  TRIGCH = APASKY(25)
      IF ( MARAKY(27) .NE. 0 )  ERRMTI = APASKY(27)
      IF ( MARAKY(35) .NE. 0 )  DTCNTI = APASKY(35)
      IF ( MARAKY(36) .NE. 0 )  RREYE2 = APASKY(36)
      IF ( MARAKY(37) .NE. 0 )  RPRNE2 = APASKY(37)
      IF ( MARAKY(38) .NE. 0 )  RSCHE2 = APASKY(38)
      IF ( MARAKY(39) .NE. 0 )  OMEGE2 = APASKY(39)
      IF ( MARAKY(40) .NE. 0 )  GFACE2 = APASKY(40)
      IF ( MARAKY(41) .NE. 0 )  CFLXTI = APASKY(41)
C
```

```
C       SOME VALUES CAN NOT BE CHANGED --- THEY MUST BE READ FROM
C       JPNTRE.DAT
C          -
C       IF ( MARAKY(24) .NE. 0 )  TIMNTI = APASKY(24)
C       IF ( MARAKY(34) .NE. 0 )  FCTRTI = APASKY(34)
        MARAKY(24) = 0
        MARAKY(34) = 0
C
C       CHANGE REFERENCE TEMPERATURE IF NEED BE
C
        IF ( MARAKY(7) .NE. 0 )  THEN
           IF (ABS(TREFFL-APASKY(7)) .GT. 1.) THEN
               CALL CHKREF
               GOTO 10
           ENDIF
        ENDIF
C
C       CHANGE REFERENCE PRESSURE IF NEED BE
C
        IF ( MARAKY(9) .NE. 0 )  THEN
           IF (ABS(PRESFL-APASKY(9)) .GT. 1.) CALL CHKREF
        ENDIF
C
C       CHANGE REFERENCE DISTANCE
C
10      IF ( MARAKY(11) .NE. 0 )  THEN
           IF (ABS(DISTFL-APASKY(11)) .GT. 0.001) THEN
              DISTFL = APASKY(11)
              WDREFL = RHORFL*UREFFL/DISTFL
           ENDIF
        ENDIF
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGE2 .NE. 2 .AND. IDBGE2 .LT. 1000)  RETURN
        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) KSRTE2, KONVE2, MITRE2, NITRE2, IDBGE2,
     1                     IDBGTI, METHA2, IDBGA2, NXTDA2, MITRA2,
     2                     IDBGFL, MALVG2, KADPTI, JREADS, KTIMTI,
     3                     NGIVTI, MITRPS, KHAFEZ, KEQNE2, KMERA2,
     4                     KFACTI, KCHKA2, IMPLTI, IDBGFR, K1ADA2,
     5                     K2ADA2, IADDH2
        WRITE(JDEBUG,1400) SMAXE2, SMINE2, CFLNTI, EPSLE2, ALPHA2,
     1                     BETAA2, GAMMA2, DELTA2, TIMXTI, TIMNTI,
     2                     EPS1TI, EPSOTI, TRIGCH, ERRMTI, FCTRTI,
     3                     DTCNTI, RREYE2, RPRNE2, RSCHE2, OMEGE2,
     4                     GFACE2

        WRITE(JDEBUG,1500) MTITLE

        WRITE(JDEBUG,1600)
        DO 20 IKEY = 1, NIPAKY
           IF (MARIKY(IKEY) .NE. 0) THEN
               WRITE(JDEBUG,1800) KYWRDI(IKEY), IPASKY(IKEY)
           ENDIF
```

```
20      CONTINUE

        WRITE(JDEBUG,1700)
        DO 30 IKEY = 1, NAPAKY
           IF (MARAKY(IKEY) .NE. 0) THEN
               WRITE(JDEBUG,1900) KYWRDA(IKEY), APASKY(IKEY)
           ENDIF
30      CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM E2RSRT' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'KSRTE2 = ',I5,15X,'KONVE2 = ',I5/
     1          5X,'MITRE2 = ',I5,15X,'NITRE2 = ',I5/
     2          5X,'IDBGE2 = ',I5,15X,'IDBGTI = ',I5/
     3          5X,'METHA2 = ',I5,15X,'IDBGA2 = ',I5/
     4          5X,'NXTDA2 = ',I5,15X,'MITRA2 = ',I5/
     5          5X,'IDBGFL = ',I5,15X,'MALVG2 = ',I5/
     6          5X,'KADPTI = ',I5,15X,'JREADS = ',I5/
     7          5X,'KTIMTI = ',I5,15X,'NGIVTI = ',I5/
     8          5X,'MITRPS = ',I5,15X,'KHAFEZ = ',I5/
     9          5X,'KEQNE2 = ',I5,15X,'KMERA2 = ',I5/
     *          5X,'KFACTI = ',I5,15X,'KCHKA2 = ',I5/
     1          5X,'IMPLTI = ',I5,15X,'IDBGFR = ',I5/
     2          5X,'K1ADA2 = ',I5,15X,'K2ADA2 = ',I5/
     3          5X,'IADDH2 = ',I5,15X,'       = ',I5/)
1400    FORMAT(5X,'SMAXE2 = ',G15.5,5X,'SMINE2 = ',G15.5/
     1          5X,'CFLNTI = ',G15.5,5X,'EPSLE2 = ',G15.5/
     2          5X,'ALPHA2 = ',G15.5,5X,'BETAA2 = ',G15.5/
     3          5X,'GAMMA2 = ',G15.5,5X,'DELTA2 = ',G15.5/
     4          5X,'TIMXTI = ',G15.5,5X,'TIMNTI = ',G15.5/
     5          5X,'EPS1TI = ',G15.5,5X,'EPSOTI = ',G15.5/
     6          5X,'TRIGCH = ',G15.5,5X,'ERRMTI = ',G15.5/
     7          5X,'FCTRTI = ',G15.5,5X,'DTCNTI = ',G15.5/
     8          5X,'RREYE2 = ',G15.5,5X,'RPRNE2 = ',G15.5/
     9          5X,'RSCHE2 = ',G15.5,5X,'OMEGE2 = ',G15.5/
     *          5X,'GFACE2 = ',G15.5                      )
1500    FORMAT(A80)
1600    FORMAT(/5X,'THE FOLLOWING INTEGER KEYS WERE ACTUALLY CHANGED')
1700    FORMAT(/5X,'THE FOLLOWING REAL KEYS WERE ACTUALLY CHANGED')
1800    FORMAT(10X,A7,2X,I7)
1900    FORMAT(10X,A7,2X,G15.6)

        RETURN
        END
```

E2SCH0

```
        SUBROUTINE E2SCH0
```

```
      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] H2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.INC     '
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] KYCOMN.INC/LIST'
C     INCLUDE '[.INC] PRCOMN.INC/LIST'
      INCLUDE '[.INC] TICOMN.INC/LIST'
      DIMENSION XVERT(2,6), DPENLH(MEQNFL,1000), X$(1000)
      CHARACTER COMAND*6, FILNAM*12, RECORD*132
      LOGICAL   IWRITE

C*********************************************************************

C     THIS SUBROUTINE FINDS THE COMMANDS AND EXECUTES THEM FOR
C     SPECIAL SITUATIONS.  FOR A NORMAL RUN THIS SUBROUTINE IS
C     NOT NEEDED.

C*********************************************************************
C
C     WANT DEBUG PRINT ?

      IWRITE  = IDBGE2 .NE. 4 .AND. IDBGE2 .LT. 1000
      IWRITE  = .NOT. IWRITE

      IF (IWRITE)  THEN
         WRITE(JDEBUG,1000)
         WRITE(JDEBUG,1100)
         WRITE(JDEBUG,1200)
         WRITE(JDEBUG,1300) JREADS
      ENDIF

C     FOR A NORMAL RUN THERE IS NO SCHEDULE PROGRAM
      IF (JREADS .EQ. 0) RETURN

C      READ THE COMMAND AND THE FILENAME (IF NECESSARY)

10    READ(JREADS,1400,END=40) COMAND, FILNAM
      IUNITS = 91

      IF (IWRITE)  THEN
         WRITE(JDEBUG,1500) COMAND, FILNAM
      ENDIF
C
C     ----------
C     PRE-EMBEDD
C     ----------
C
C     DO THE PRE-EMBEDDED OF GRIDS ACCORDING TO INITIAL CONDITIONS
C1    I.E., ADAPATATION BEFORE INTEGRATION
C
```

```
      IF (COMAND .EQ. 'PREEMB') THEN
C          -
C        INPUT THE INTERPOLATION INDICATOR AS NEGATIVE VALUE IF
C        SPECIAL INTERPOLATION IS DESIRED FOR DIVIDED CELLS,
C        THIS MAY BE USEFUL WITH PROBLEMS WHERE A STEP FUNCTION
C        IS INTRODUCED AS AN I.C. NOTE THAT THE SHOCK IS REGARDED
C        TO BE A PART OF LEFT HAND REGION AND CORRECTION IS ONLY
C        MADE ON RHS.
         READ(JREADS,*) NVERT1
C        INPUT THE NUMBER OF CELLS TO BE EXTENDED; IT IS DESIRABLE
C        TO HAVE LARGER EXTENSION HERE;  IF THE SAME NUMBER OF
C        CELL EXTENSION IS TO BE USED THEN INPUT ZERO
         READ(JREADS,*) NXTD
         NDUMMY = NXTDA2
         IF (NXTD .GT. 0) NXTDA2 = NXTD
         NCELP  = NCELG2
C        NOW PRE-EMBEDD
         CALL A2MTHO
C        RESET THE EXTENSION CELL NUMBER
         NXTDA2 = NDUMMY
         IF (IWRITE) THEN
            WRITE(JDEBUG,1600) NVERT
            WRITE(JDEBUG,1700) NXTD, NXTDA2
         ENDIF
C
C                                FROZEN OR EQUILIBRIUM SHOCK
C
C      ---*-----*-----*-----*      ____________________
C                          |      |         |          |
C                          |      | JCELL   | NB1      |
C                          |      |_________|__________|
C                          |    IEDGE1     IEDGE2    IOUT
C                          |
C                          |----*-----*-----*---
C
C
C
C        CORRECT THE INTERPOLATION IF NEED BE
         DO 30 ICELL = NCELP+1, NCELG2, 4
C           FIND THE SUPERCELL
            JCELL = ICELG2(10,ICELL)
            IF (NVERT1 .LT. 0 .AND. JCELL .GT. 0) THEN
               IEDGE2 = ICELG2(4,JCELL)
               NB1    = NEIBG2(3,IEDGE2)
               IF (NB1 .NE. 0) THEN
                 IEDGE1 = ICELG2(2,JCELL)
                 IOUT   = ICELG2(4,NB1)
                 IF(DPENG2(1,IEDGE1).EQ.DPENG2(1,IEDGE2))GOTO30
                 IF (DPENG2(1,IEDGE2).EQ.DPENG2(1,IOUT)) THEN
                   PRESG2(ICELG2(1,JCELL)) = PRESG2(IOUT)
                   PRESG2(ICELG2(3,JCELL)) = PRESG2(IOUT)
                   PRESG2(ICELG2(7,JCELL)) = PRESG2(IOUT)
                   TEMPG2(ICELG2(1,JCELL)) = TEMPG2(IOUT)
                   TEMPG2(ICELG2(3,JCELL)) = TEMPG2(IOUT)
                   TEMPG2(ICELG2(7,JCELL)) = TEMPG2(IOUT)
                   DO 20 I = 1, NEQNFL
                     DPENG2(I,ICELG2(1,JCELL)) = DPENG2(I,IOUT)
```

```
                  DPENG2(I,ICELG2(3,JCELL)) = DPENG2(I,IOUT)
                  DPENG2(I,ICELG2(7,JCELL)) = DPENG2(I,IOUT)
20              CONTINUE
                ENDIF
              ENDIF
            ENDIF
30        CONTINUE
C         SEE IF DEBUG CHECK IS NEEDED
          IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
            NERR = 0
            CALL CHKBN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
            CALL CHKNC2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
            CALL CHKNN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
            CALL CHKSP2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
          ENDIF
C
        ENDIF
C
C       --------------------
C       CHEMISTRY INPUT HELP
C       --------------------
C
C2      HELP IN WRITTING THE FILE INPUTC.DAT
C
        IF (COMAND .EQ. 'C2HELP') CALL C2HELP (IUNITS)
C
C       ----------
C       END OF RUN
C       ----------
C
C3      FINISH THIS RUN

40      IF (COMAND .EQ. 'FINISH') CALL E2FINI
C
C       ----------------------
C       SPECIFIC GRID DIVISION
C       ----------------------
C
C4      DIVIDE THE GRID FOR SPECIFIED CELLS

        IF (COMAND .EQ. 'G2DIVO') THEN
          READ(JREADS,*) NOCELL
          IF (IWRITE) WRITE (JDEBUG,1800) NOCELL
          DO 50 I = 1, NOCELL
            READ(JREADS,*) ICELL
            IF (IWRITE) WRITE (JDEBUG,1900) I, ICELL
            IWARN = 0
            CALL G2DIVO (ICELL, IWARN)
            IF (IWARN .NE. 0) WRITE(JTERMO,2000) IWARN, ICELL
C           SEE IF DEBUG CHECK IS NEEDED
            IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
              NERR = 0
              CALL CHKBN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
              CALL CHKNC2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
              CALL CHKNN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
              CALL CHKSP2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
            ENDIF
```

```
50        CONTINUE
C         SET THE CEWIC CELL POINTERS
          CALL A2CEWC
         ENDIF
C
C        -----------------
C        PRINT GRID DETAILS
C        -----------------
C
C5       PRINT THE DETAILS OF GRIDS
         IF (COMAND .EQ. 'G2PRNT') THEN
C          READ THE OPTION PARAMETER
           READ(JREADS,*) IOPTGP
           CALL G2PRNT (IOPTGP)
         ENDIF
C
C        -----------------
C        PRINT GRID SUMMARY
C        -----------------
C
C6       WRITE A SUMMARY OF GRIDS
         IF (COMAND .EQ. 'G2SUMY') THEN
            WRITE(IUNITS,2100)
            CALL  G2SUMY (IUNITS)
            CLOSE (IUNITS)
         ENDIF
C
C7       WRITE THE OUTPUT FOR SUBSEQUENT PLOTTING AT VARIOUS TIME
C        STATIONS; ALSO OPEN THE UNIT WHERE THE OUTPUT WILL BE
C        WRITTEN.  IF THE FILE IS OLD (RESTARTED CASE) GOTO THE
C        END OF THE FILE
C
         IF (COMAND .EQ. 'G2TIME') THEN
           IF (KTIMTI .NE. 2) KTIMTI = 1
           TIME   = TIMNTI
C          IF (KSRTE2 .EQ. 0 .OR. KTIMTI .EQ. 2) THEN
C            OPEN (UNIT=JCARDS, FILE='G2TIME.DAT', STATUS='NEW')
C            CALL   PSSUMY
C          ELSE
             OPEN (UNIT=JCARDS, FILE='G2TIME.DAT', STATUS='OLD')
60           READ (JCARDS, 2200, END=70) FILNAM
             GO TO 60
C          ENDIF
70         CALL   G2TIME (TIME, 1)
         ENDIF
C
C        ---------------------
C        REGIONAL GRID DIVISION
C        ---------------------
C
C8       DIVIDE THE GRIDS IN THE SPECIFIED POLYGONAL REGION
C        MANUAL SPATIAL EMBEDDING (MAXIMUM POLYGON : HEXAGON)

         IF (COMAND .EQ. 'MANUAL') THEN
C          READ THE NUMBER OF VERTICES OF THE POLYGONAL REGION,
C          INPUT NEGATIVE VALUE IF SPECIAL INTERPOLATION IS
C          DESIRED FOR DIVIDED CELLS,  THIS MAY BE USEFUL WITH
```

```
C         PROBLEMS WHERE A STEP FUNCTION IS INTRODUCED AS AN I.C.
C         NOTE THAT THE SHOCK IS REGARDED TO BE A PART OF LEFT
C         HAND REGION AND CORRECTION IS ONLY MADE ON RHS.
          READ(JREADS,*) NVERT1
          NVERT = ABS(NVERT1)
          IF (IWRITE) WRITE(JDEBUG,2300) NVERT
C         NOW READ THE COORDINATES OF THESE VERTICES
          DO 80 IVERT = 1, NVERT
            READ(JREADS,*) XVERT(1,IVERT), XVERT(2,IVERT)
            IF (IWRITE) WRITE(JDEBUG,2400) IVERT,
     1                        XVERT(1,IVERT), XVERT(2,IVERT)
80        CONTINUE
C         INITIALIZE THE NUMBER OF CELLS TO BE DIVIDED
          NCELLD = 0
C         LOOP THROUGH ALL THE CEWIC CELLS; A CELL WILL BE DIVIDED IF THE
C         CENTER OF THE CELL LIES WITHIN THE SPECIFIED POLYGON
          DO 90 ICELL = 1, NCELG2
             KC = ICELG2(1,ICELL)
             IF (KC .EQ. 0) THEN
               KSW   = ICELG2 (2, ICELL)
               KSE   = ICELG2 (4, ICELL)
               KNE   = ICELG2 (6, ICELL)
               KNW   = ICELG2 (8, ICELL)
               XSW   = GEOMG2 (1, KSW  )
               XSE   = GEOMG2 (1, KSE  )
               XNW   = GEOMG2 (1, KNE  )
               XNW   = GEOMG2 (1, KNW  )
               YSW   = GEOMG2 (2, KSW  )
               YSE   = GEOMG2 (2, KSE  )
               YNW   = GEOMG2 (2, KNE  )
               YNW   = GEOMG2 (2, KNW  )
               XC    = 0.25*(XSW + XSE + XNE + XNW)
               YC    = 0.25*(YSW + YSE + YNE + YNW)
               IIN   = 0
               CALL INSIDE (IIN, XVERT, NVERT, XC, YC)
C              MARK THE NODE IF THE CELL IS IN THIS REGION
               IF (IIN .EQ. 1) THEN
                    NCELLD         = NCELLD + 1
                   MRKDA2(NCELLD) = KSW
               ENDIF
             ENDIF
90        CONTINUE
C         DEBUG WRITE
          IF (IWRITE) THEN
             WRITE(JDEBUG,2500) NCELLD
             WRITE(JDEBUG,2600) (NEIBG2(3,MRKDA2(JN)),JN=1,NCELLD)
          ENDIF
C         CALL THE GRID DIVIDE ROUTINE FOR ALL THE PREVIOUSLY
C         COLLECTED CELLS.
          DO 120 JNODE = NCELLD, 1, -1
             KSW   = MRKDA2  (JNODE)
             JCELL = NEIBG2 (3,KSW)
             IWARN = 0
             CALL    G2DIVO (JCELL, IWARN)
             IF (IWARN .NE. 0) WRITE(JTERMO,2000) IWARN, JCELL
C            CORRECT THE INTERPOLATION IF NEED BE
             IF (NVERT1 .LT. 0) THEN
```

```
                NB1 = NEIBG2(3,ICELG2(4,JCELL))
                IF (NB1 .NE. 0) THEN
                  IEDGE1 = ICELG2(2,JCELL)
                  IEDGE2 = ICELG2(4,JCELL)
                  IOUT   = ICELG2(4,NB1)
                  IF(DPENG2(1,IEDGE1).EQ.DPENG2(1,IEDGE2))GOTO110
                  IF (DPENG2(1,IEDGE2).EQ.DPENG2(1,IOUT)) THEN
                    PRESG2(ICELG2(1,JCELL)) = PRESG2(IOUT)
                    PRESG2(ICELG2(3,JCELL)) = PRESG2(IOUT)
                    PRESG2(ICELG2(7,JCELL)) = PRESG2(IOUT)
                    TEMPG2(ICELG2(1,JCELL)) = TEMPG2(IOUT)
                    TEMPG2(ICELG2(3,JCELL)) = TEMPG2(IOUT)
                    TEMPG2(ICELG2(7,JCELL)) = TEMPG2(IOUT)
                    DO 100 I = 1, NEQNFL
                      DPENG2(I,ICELG2(1,JCELL)) = DPENG2(I,IOUT)
                      DPENG2(I,ICELG2(3,JCELL)) = DPENG2(I,IOUT)
                      DPENG2(I,ICELG2(7,JCELL)) = DPENG2(I,IOUT)
100                  CONTINUE
                    ENDIF
                  ENDIF
                ENDIF
C               SEE IF DEBUG CHECK IS NEEDED
110             IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
                  NERR = 0
                  CALL CHKBN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                  CALL CHKNC2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                  CALL CHKNN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                  CALL CHKSP2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
                ENDIF
120           CONTINUE
C             REMOVE THE VOIDS
              CALL A2VOID
C             SET THE CEWIC CELL POINTERS
              CALL A2CEWC
            ENDIF
C
C           ----------
C           NORMAL RUN
C           ----------
C
C9          DO THE "NORMAL" RUN

            IF (COMAND .EQ. 'NORMAL') THEN
              READ(JREADS,*) NIT
              IF (NIT .NE. 0) MITRE2 = NIT
              RETURN
            ENDIF
C
C           -------------
C           PRINT RESULTS
C           -------------
C
C10         PRINT ALL THE RESULTS

            IF (COMAND .EQ. 'PRINTO') THEN
C             SET THE PRINTOUT PARAMETER
              KPRINT = NINT(APASKY(19))
```

```
            CALL WRINI2
            IF (KPRINT .EQ. 2) THEN
             WRITE(JOUTAL,2700)
             WRITE(JOUTAL,2800) NITRE2, NNODG2, NCELG2, NBNDG2, NCELA2
             WRITE(JOUTAL,2900) TIMNTI
            ENDIF
            IF (KPRINT .GT. 2) CALL G2RESO
          ENDIF
C
C         --------------------
C         WRITE POINTER SYSTEM
C         --------------------
C
C11       WRITE THE WHOLE POINTER SYSTEM ON THE SPECIFIED FILE

          IF (COMAND .EQ. 'PSWRT2') THEN
            IF (KSRTE2 .LT. 1000) THEN
              CALL  PSWRT2 (JPNTWR)
            ELSE
              CALL  PSWRTU (JPNTWR)
            ENDIF
            CLOSE (JPNTWR)
          ENDIF
C
C         ---------------
C         CHANGE B.C. TYPE
C         ---------------
C
C12       CHANGE THE TYPE OF BOUNDARY CONDITIONS AT THE SPECIFIED BOUNDARY
          IF (COMAND .EQ. 'CHNBND') THEN
C           READ THE SPECIFIC BOUNDARY (SURFACE) WHERE CHANGE IS DESIRED
C           ISURFC=3 FOR SOUTH; 5 FOR EAST; 7 FOR NORTH; 9 FOR WEST
            READ(JREADS,*) ISURFC
C           READ THE OLD AND NEW BOUNDARY TYPES
            READ(JREADS,*) IBCOLD, IBCNEW
            DO 130 INB = 1, NBNDG2
              IF (IBNDG2(4,INB) .EQ. ISURFC) THEN
                IF (IBNDG2(5,INB) .EQ. IBCOLD) THEN
                  WRITE(JTERMO,2950) INB, ISURFC, IBCOLD, IBCNEW
                  IBNDG2(5,INB) = IBCNEW
                ENDIF
              ENDIF
130         CONTINUE
          ENDIF
C
C         ----------------------
C         CHANGE DISSOCIATION PHI
C         ----------------------
C
C13       CHANGE DISSOCIATION PHI IN THE MIDDLE OF A RESTART CASE
          IF (COMAND .EQ. 'DISPHI') THEN
C           SEE IF THE APPROPRIATE MODEL IS BEING USED
            IF (KROGER .NE. 2) GO TO 10
            PHIOLD = CHNGE2(1,1)
            RHOD   = CHNGE2(1,2)
            IF (PHIOLD .EQ. 0.) THEN
              PHIOLD = APASKY(1)
```

```
            RHOD   = APASKY(2)
          ENDIF
          write(6,*) ' phiold=',phiold
          write(6,*) ' rhod  =',rhod
C         READ THE NEW PHI
          READ(JREADS,*) PHI
          CHNGE2(1,1) = PHI
          ETA         = EXPFCH(1)
          THETD       = ENEFCH(1)
          TOETA       = TREFFL**ETA
          UNITCF      = UREFFL/(TOETA*RHORFL*DISTFL)
          CFBMA       = UNITCF*PHI
          CF          = CFBMA*AMWTCH(1)
          PREFCH(1)   = LOG(CFBMA)
          PREBCH(1)   = LOG(0.5*CF/RHOD)
          APASKY(1)   = PHI
        ENDIF
C
C       --------------
C       CHECK POINTERS
C       --------------
C
C14     CHECK THE POINTER SYSTEM GLOBALLY
C
        IF (COMAND .EQ. 'CHKPNT') THEN
           NERR = 0
           CALL CHKBN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
           CALL CHKNC2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
           CALL CHKNN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
           CALL CHKSP2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
        ENDIF
C
C       ----------------
C       SMOOTH THE NODES
C       ----------------
C
C15     CHECK ALL THE NODES FOR KINKS OR OSCILLATIONS
C
        IF (COMAND .EQ. 'G2SMOT') CALL G2SMOT
        IF (COMAND .EQ. 'G3SMOT') CALL G3SMOT
        IF (COMAND .EQ. 'A2CEWC') CALL A2CEWC
        IF (COMAND .EQ. 'TVINIO') CALL TVINIO
        IF (COMAND .EQ. 'ROGERC') CALL ROGERC
        IF (COMAND .EQ. 'G4SMOT') THEN
C           READ THE VARIABLE WHICH IS TO BE SMOOTHENED
            READ(JREADS,*) IT
            CALL G4SMOT(IT)
            IF (IWRITE) WRITE (JDEBUG,*) ' goes to g4smot'
        ENDIF
        IF (COMAND .EQ. 'E2DAMP') CALL E2DAMP
C
C       -------------------------------------------
C       INTERPOLATION FOR BOUNDARY NODE CELLS - REGION
C       -------------------------------------------
C
C16     SET THE BOUNDARY NODE CELLS IN A REGION (WHOLE SURFACE) FOR
C       SPECIAL INTERPOLATION FUNCTIONS
```

```
C
C       SBNCRG :- Set Boundary Node Cell in ReGion
C
        IF (COMAND .EQ. 'SBNCRG') THEN
          WRITE(JTERMO,3000)
C         READ THE SPECIFIC BOUNDARY (SURFACE) WHERE CHANGE IS DESIRED
C         ISURFC=3 FOR SOUTH; 5 FOR EAST; 7 FOR NORTH; 9 FOR WEST
          READ(JREADS,*) ISURFC
C         READ THE NEW INTERPOLATION FUNCTION INDICATOR INTERF
C         INTERF=1 FOR QUADRATIC; =2 FOR CUBIC; =3 FOR CIRCULAR ARC
          READ(JREADS,*) INTERF
          IF (ISURFC .EQ. 7) ISURFC = 12
C         ONLY NORTH AND SOUTH BOUNDARIES CAN BE ADJUSTED HERE
          IF (.NOT.(ISURFC .EQ. 3 .OR. ISURFC .EQ. 12)) GOTO 10
C         SET THE EDGE POINTERS FOR THE BOUNDARY CELLS
          IF (ISURFC .EQ. 3) THEN
            IEDGE1 = 2
            IEDGE2 = 4
          ELSE
            IEDGE1 = 6
            IEDGE2 = 8
          ENDIF
C         SET THE THIRD BYTE INTEGER FOR IOR FUNCTION
          KNTERF = 0
          IF (INTERF .EQ. 1) KNTERF = KLO100
          IF (INTERF .EQ. 2) KNTERF = KLO200
          IF (INTERF .EQ. 3) KNTERF = KLO300
C
          DO 140 ICELL = 1, NCELG2
             KX    = KAUXG2(ICELL)
             KTEST = IAND(KX,KLOOOF)
C            CHECK IF ON THE CORRECT BOUNDARY SURFACE
             IF (KTEST .EQ. ISURFC) THEN
               X1 = GEOMG2(1,ICELG2(IEDGE1,ICELL))
               X2 = GEOMG2(1,ICELG2(IEDGE2,ICELL))
               Y1 = GEOMG2(2,ICELG2(IEDGE1,ICELL))
               Y2 = GEOMG2(2,ICELG2(IEDGE2,ICELL))
C              NO CHANGE IS REQUIRED FOR HORIZONTAL OR VERTICAL BOUNDARIES
               DXTEST = ABS(X1-X2)
               IF (DXTEST .LT. 1.E-8) GOTO 140
               DXTEST = ABS(Y1-Y2)
               IF (DXTEST .LT. 1.E-8) GOTO 140
C              OVERLAY THE THIRD BYTE ONTO KAUXG2
               KAUXG2(ICELL) = IOR(KAUXG2(ICELL),KNTERF)
               WRITE(JTERMO,3100) ICELL, KX, KAUXG2(ICELL), X1,Y1,X2,Y2
             ENDIF
140       CONTINUE
        ENDIF
C
C       ------------------------------------------
C       INTERPOLATION FOR A SINGLE BOUNDARY NODE CELL
C       ------------------------------------------
C
C17     SET A GIVEN BOUNDARY NODE CELL FOR SPECIAL INTERPOLATION
C       FUNCTIONS
C
C       SBNCIN :- Set Boundary Node Cell INdividually
```

```
C
      IF (COMAND .EQ. 'SBNCIN') THEN
        WRITE(JTERMO,3000)
C       READ THE SPECIFIC CELL UNDER CONSIDERATION
        READ(JREADS,*) ICELL
C       READ THE NEW INTERPOLATION FUNCTION INDICATOR INTERF
C       INTERF=1 FOR QUADRATIC; =2 FOR CUBIC; =3 FOR CIRCULAR ARC
        READ(JREADS,*) INTERF
        KX    = KAUXG2(ICELL)
        KTEST = IAND(KX,KL000F)
C       ONLY NORTH AND SOUTH BOUNDARIES CAN BE ADJUSTED HERE
        IF (.NOT.(KTEST .NE. 3 .OR. KTEST .NE. 12)) GOTO 10
C       SET THE EDGE POINTERS FOR THE BOUNDARY CELLS
        IF (KTEST .EQ. 3) THEN
          IEDGE1 = 2
          IEDGE2 = 4
        ELSE
          IEDGE1 = 6
          IEDGE2 = 8
        ENDIF
C       SET THE THIRD BYTE INTEGER FOR IOR FUNCTION
        KNTERF = 0
        IF (INTERF .EQ. 1) KNTERF = KL0100
        IF (INTERF .EQ. 2) KNTERF = KL0200
        IF (INTERF .EQ. 3) KNTERF = KL0300
C
        X1 = GEOMG2(1,ICELG2(IEDGE1,ICELL))
        X2 = GEOMG2(1,ICELG2(IEDGE2,ICELL))
        Y1 = GEOMG2(2,ICELG2(IEDGE1,ICELL))
        Y2 = GEOMG2(2,ICELG2(IEDGE2,ICELL))
C       NO CHANGE IS REQUIRED FOR HORIZONTAL OR VERTICAL BOUNDARIES
        DXTEST = ABS(X1-X2)
        IF (DXTEST .LT. 1.E-8) GOTO 10
        DXTEST = ABS(Y1-Y2)
        IF (DXTEST .LT. 1.E-8) GOTO 10
C       OVERLAY THE THIRD BYTE ONTO KAUXG2
        KAUXG2(ICELL) = IOR(KAUXG2(ICELL),KNTERF)
        WRITE(JTERMO,3100) ICELL, KX, KAUXG2(ICELL), X1,Y1,X2,Y2
      ENDIF
C
C     --------------------------------
C     SAVE PREVIOUS CONVERGENCE HISTORY
C     --------------------------------
C
C18   FOR A RESTART CASE IN WHICH CONVERGENCE HISTORY IS IMPORTANT,
C     THE CALCULATIONS OF THE PREVIOUS RUN MUST BE APPENDED TO THE
C     PRESENT CASE
C
      IF (COMAND .EQ. 'SHISTO') THEN
        IF (KSRTE2 .EQ. 0 .OR. KSRTE2 .EQ. 1000) GOTO 10
        OPEN (UNIT=JDUMY4, FILE='JHISTO.DAT;-1', STATUS='OLD')
C       READ THE PREVIOUS TITLE AND DISCARD IT
        READ (JDUMY4, 3200, END=160) RECORD
C       READ THE REST OF THE FILE AND KEEP IT
150     READ (JDUMY4, 3200, END=160) RECORD
        WRITE(JHISTO, 3200) RECORD
        GOTO 150
```

```
160       CLOSE (JDUMY4)
         ENDIF
C            -
C        -------------------
C        CHANGE FORMAT PSWRIT
C        -------------------
C
C19      CHANGE THE FORMAT OF THE POINTER SYSTEM FILE (PSRED2 OR PSREDU)

         IF (COMAND .EQ. 'PSCHAN') THEN
C          INPUT THE VARIABLE NVERT1 TO INDICATE IF
C            1. TO CHANGE FROM FORMATTED TO UNFORMATTED FORM
C            2. TO CHANGE FROM UNFORMATTED TO FORMATTED FORM
           READ(JREADS,*) NVERT1
           ZCUM = WORKA2(3)
           WRITE (6,*) ' ZCUM IN WORKA2(3) = ',ZCUM
           MRKDA2(3) = -99
           IF (NVERT1 .EQ. 1) THEN
             CALL  PSWRTU (JPNTWR)
           ELSE
             CALL  PSWRT2 (JPNTWR)
           ENDIF
           CLOSE (JPNTWR)
           STOP ' THE END'
         ENDIF
C
C        --------------------------------
C        ADD FUEL AT A GIVEN PLANE LOCATION
C        --------------------------------
C
C20      THIS PROCEDURE INJECTS THE FUEL AT A GIVEN VERTICAL PLANE,
C        ONLY THE LOWERMOST POINT OF THE PLANE IS NEEDED

         IF (COMAND .EQ. 'FUELH2') THEN
C          INPUT THE INITIAL POINT OF INJECTION ON THE PLANE
           READ(JREADS,*) ISTART
C          INPUT THE EQUIVALENCE RATIO FOR THE FUEL ADDITION
           READ(JREADS,*) PHICR
           IBASH2  = ISTART
           PHIEH2  = PHICR
           CALL H2INIT
         ENDIF
c
         IF (COMAND .EQ. 'FUELH3') THEN
           CALL H3INIT
         ENDIF
         IF (COMAND .EQ. 'SCREEN') THEN
           CALL H2SCRI
         ENDIF
         IF (COMAND .EQ. 'PUTSCR') THEN
           CALL H2SCRN
         ENDIF
         IF (COMAND .EQ. 'CHKMAS') THEN
           CALL CHKMAS
         ENDIF
         IF (COMAND .EQ. 'PUTH2I') THEN
           CALL H2TRIN
```

```
      ENDIF
      IF (COMAND .EQ. 'H2FLOT') THEN
        CALL H2FLOT
      ENDIF
      IF (COMAND .EQ. 'HSHEAR') THEN
        CALL HSHEAR
      ENDIF
      IF (COMAND .EQ. 'PSWCOR') THEN
        CALL PSWCOR(JPNTWR)
      ENDIF
      IF (COMAND .EQ. 'CHANKE') THEN
C       READ THE REACTION NUMBER FOR WHICH THE CHANGE OF KE IS DESIRED
        READ(JREADS,*) IR
        WRITE(6,*) ' OLD KE',PREECH(IR)
        READ(JREADS,*) PREECH(IR)
        WRITE(6,*) ' NEW KE',PREECH(IR)
      ENDIF
      IF (COMAND .EQ. 'CHKTM2') THEN
        DO 165 INODE = 1, NNODG2
          IF (PRESG2(INODE).LT.0. .OR. TEMPG2(INODE).LT.0.)
     1                CALL CHKTM2(INODE)
165     CONTINUE
      ENDIF
C
C     -------------
C     PRE-EMBEDD II
C     -------------
C
C     DO THE PRE-EMBEDDED OF GRIDS ACCORDING TO INITIAL CONDITIONS
C21   I.E., ADAPATATION BEFORE INTEGRATION;  FOR NON-EQUILIBRIUM
C     SHOCKS THE INTERPOLATION TYPE OF 'PREEMB' DOES NOT HOLD SINCE
C     THE SHOCK IS NO LONGER A STEP FUNCTION.  LINEAR INTERPOLATION
C     IS DONE WHICH IS BASED UPON A PREVIOUSLY WRIITEN FILE WITH
C     FINE X-STEP SIZE.  THE INTERPOLATION IS VALID ONLY FOR STRAIGHT
C     CHANNELS.  THE NAME OF OLD FILE IS LHSHOC.INT
C
      IF (COMAND .EQ. 'PREEM2') THEN
C       INPUT THE NUMBER OF CELLS TO BE EXTENDED; IT IS DESIRABLE
C       TO HAVE LARGER EXTENSION HERE;  IF THE SAME NUMBER OF
C       CELL EXTENSION IS TO BE USED THEN INPUT ZERO
        READ(JREADS,*) NXTD
        NDUMMY = NXTDA2
        IF (NXTD .GT. 0) NXTDA2 = NXTD
        NCELP  = NCELG2
C       NOW PRE-EMBEDD
        CALL A2MTHO
C       RESET THE EXTENSION CELL NUMBER
        NXTDA2 = NDUMMY
        IF (IWRITE) THEN
          WRITE(JDEBUG,1600) NVERT
        ENDIF
C
C                                  NON-EQUILIBRIUM SHOCK
C
C     ---*-----*-----*---          ---------------------
C                        |         |         |         |
C                        |         | JCELL   | NB1     |
```

```
C                          |      |_________|_________|
C                          |    IEDGE1     IEDGE2    IOUT
C           -              |
C                          *-----*-----*-----*---
C
C
C
C         READ THE ADDITIONAL GRID INFORMATION FROM THEUNFORMATTED FILE

          OPEN (UNIT=58, FILE='LHSHOC.INT', STATUS='OLD',
     1               FORM='UNFORMATTED', READONLY)
          REWIND (58)
          READ(58) NPOINT, NTOTAL

          DO 210 IP = 1, NPOINT
              READ (58) X$(IP),(DPENLH(K,IP), K = 1, NTOTAL)
210       CONTINUE

C         CORRECT THE INTERPOLATION IF NEED BE

          DO 240 ICELL = NCELP+1, NCELG2, 4
C            FIND THE SUPERCELL
             JCELL = ICELG2(10,ICELL)
             IF (JCELL .GT. 0) THEN
                IEDGE1 = ICELG2(2,JCELL)
                IEDGE2 = ICELG2(4,JCELL)
                IF(DPENG2(1,IEDGE1).EQ.DPENG2(1,IEDGE2))GOTO240
                XI = GEOMG2(1,ICELG2(1,JCELL))
                DO 230 IP = 1, NPOINT-1
                  IF (XI .GE. X$(IP+1) .AND. XI .LE. X$(IP)) THEN
                     XRAT = (XI-X$(IP))/(X$(IP+1)-X$(IP))
                     DO 220 IQ = 1, NEQNFL
                       DELTAA = DPENLH(IQ,IP+1) - DPENLH(IQ,IP)
                       ALPHA  = DPENLH(IQ,IP) + DELTAA*XRAT
                       DPENG2(IQ,ICELG2(1,JCELL)) = ALPHA
                       DPENG2(IQ,ICELG2(3,JCELL)) = ALPHA
                       DPENG2(IQ,ICELG2(7,JCELL)) = ALPHA
C                      CORRECT THE PRESSURE AND TEMPERATURE
                       CALL E2PRMT (ICELG2(1,JCELL),1)
                       CALL E2PRMT (ICELG2(3,JCELL),1)
                       CALL E2PRMT (ICELG2(7,JCELL),1)
220                  CONTINUE
                     GOTO 240
                  ENDIF   ! XI TEST
230               CONTINUE
               ENDIF        ! SUPERCELL EXITS
240       CONTINUE
C         SEE IF DEBUG CHECK IS NEEDED
          IF (IAND(KCHKA2,KL0001) .NE. 0) THEN
             NERR = 0
             CALL CHKBN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
             CALL CHKNC2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
             CALL CHKNN2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
             CALL CHKSP2 (JCELL, 0, 0, 0, 0, NERR, 'AFTDIV')
          ENDIF
C
        ENDIF  !  END OF TEST
```

```
C
C
C       GO BACK FOR MORE COMMANDS

        GO TO 10

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM E2SCHO' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'JREADS = ',I7)
1400    FORMAT(A6,4X,A12)
1500    FORMAT(5X,'COMAND = ',A8,10X,'FILNAM = ',A12)
1600    FORMAT(5X,'INTERPOLATION INDICATOR',I5)
1700    FORMAT(5X,'EXTENSION CELLS (NEW AND OLD)',2I5)
1800    FORMAT(5X,'# OF CELLS TO BE DIVIDED :',I5)
1900    FORMAT(5X,'CELL',I5,2X,' TO BE DIVIDED :',I5)
2000    FORMAT(5X,'WARNING #',I3,2X,'ISSUED FOR CELL',I5)
2100    FORMAT(' ****** SUMMARY OF GRIDS ******'/)
2200    FORMAT(A12)
2300    FORMAT(5X,'SPATIAL EMBEDDING IN POLYGON OF SIDES :',I2)
2400    FORMAT(5X,'VERTEX :',I2,7X,'X =',G14.5,10X,'Y =',G14.5)
2500    FORMAT(5X,'NO. OF CELLS TO BE DIVIDED :',I5/
     1         5X,'THE CELLS TO BE DIVIDED ARE :'/)
2600    FORMAT(20I5)
2700    FORMAT('1'//)
2800    FORMAT(5X,'TOTAL NUMBER OF ITERATIONS     = ',I5,10X,
     1         5X,'TOTAL NUMBER OF NODES          = ',I5 /
     2         5X,'TOTAL NUMBER OF CELLS          = ',I5 ,10X,
     3         5X,'TOTAL NUMBER OF BOUNDARY NODES = ',I5 /
     4         5X,'TOTAL NUMBER OF CEWIC CELLS    = ',I5 /)
2900    FORMAT(5X,'TIME =',G14.5)
2950    FORMAT(5X,'IBN=',I5,5X,'ISURFC='I5,5X,'IBCOLD=',I5,
     1         5X,'IBCNEW=',I5)
3000    FORMAT(4X,'ICELL',9X,'KX',6X,'KXNEW',6X,'X1',8X,
     1         'Y1',8X,'X2',8X,'Y2')
3100    FORMAT(2X,I6,2X,2Z10,4F10.3)
3200    FORMAT(A)

        END
```

E2SOLF

```
        SUBROUTINE E2SOLO (ITGL)
C                 E2SOLF

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'CHCOMN.INC'
```

```
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'HEXCOD.INC'
      INCLUDE 'JACOMN.INC'
      INCLUDE 'M2COMN.INC'
      INCLUDE 'PRCOMN.INC'
      INCLUDE 'TICOMN.INC'
      COMMON/WUCOMN/ WUJACO
      DIMENSION BIGFS (MEQNFL)         ,   BIGFE (MEQNFL)         ,
     1          BIGFN (MEQNFL)         ,   BIGFW (MEQNFL)         ,
     2          BIGGS (MEQNFL)         ,   BIGGE (MEQNFL)         ,
     3          BIGGN (MEQNFL)         ,   BIGGW (MEQNFL)         ,
     4          DELSW (MEQNFL)         ,   DELSE (MEQNFL)         ,
     5          DELNW (MEQNFL)         ,   DELNE (MEQNFL)         ,
     6          BWCELL(MEQNFL)         ,   DPENFA(MEQNFL,4)       ,
     7          FUJACO(MEQNFL,MEQNFL) ,   GUJACO(MEQNFL,MEQNFL) ,
     8          WUJACO(MEQNFL,MEQNFL) ,   DUCELL(MEQNFL)

      DIMENSION UTOP  (MEQNFL)         ,   TOTAL (MEQNFL)         ,
     1          WTOP  (MEQNFL)         ,   WBOT  (MEQNFL)

      DATA FUJACO /100*0./
      DATA GUJACO /100*0./
      DATA WUJACO /100*0./
      DATA BWCELL /10*0./

C*********************************************************************

C       THIS SUBROUTINE STEPS THROUGH EACH CELL ON THE SPATIAL LEVEL ITGL
C       AND APPLIES NI'S SCHEME, I.E., INTEGRATES OVER ALL CELLS ON ITGL.
C       IT ALSO COMPUTES THE ANALYTICAL AS WELL NUMERICAL JACOBIANS,
C       BECAUSE THEIR STORAGE IS COSTLY.  THIS SUBROUTINE CAN BE USED
C       FOR GRIDS WHICH HAVE NOT BEEN EMBEDDED YET.
C       DPENFA : VALUES OF DEPENDENT VARIABLES AT THE FACES
C       DPENG2 : VALUES OF DEPENDENT VARIABLES AT THE NODES
C       DPENJA : VALUES OF DEPENDENT VARIABLES FOR COMPUTING JACOBIANS
C       !!!!!!    THIS SUBROUTINE IS SPECIALIZED FOR MEQNFL=10     !!!!!!

C*********************************************************************
C
C       IMPLTI = 0 MEANS DO IMPLICIT SOURCE TERMS
C                1 MEANS DO EXPLICIT SOURCE TERMS
C                2 MEANS DO EXPLICIT SOURCE TERMS WITH FROZEN CHEMISTRY

        GOTO (310,10,610) IMPLTI+1
C       RETURN

C       USE EXPLICIT SOURCE TERMS
C       STEP THROUGH EACH CELL AT THIS LEVEL
C
CVD$    NOLSTVAL
10      DO 160 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C          ----------------------
C          CELL/NODE DETERMINATION
C          ----------------------
```

```
C          FIND THE CELL TO BE INTEGRATED
           -
           ICELL = ICELTI(JCELL)

C          SET UP NODE POINTERS FOR THIS CELL

           KSW = ICELG2( 2,ICELL)
           KSE = ICELG2( 4,ICELL)
           KNE = ICELG2( 6,ICELL)
           KNW = ICELG2( 8,ICELL)

C          --------
C          GEOMETRY
C          --------
C
C          GEOMETRY OF ALL CELL CORNERS

           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)

           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)

           XNE = GEOMG2(1,KNE)
           YNE = GEOMG2(2,KNE)

           XNW = GEOMG2(1,KNW)
           YNW = GEOMG2(2,KNW)
C
C          THE RATIO DELTA-t TO CELL VOLUME
           DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

C          COMPUTE THE AVERAGE COEFFICIENT FOR DIFFUSION

           DSDIFF = 0.5*PERIM2(ICELL)*DTDVOL

C          -------------
C          FACIAL VALUES
C          -------------
C
C          COMPUTE THE DEPENDENT VARIABLES AT THE FACES

           PRESSS = 0.5*( PRESG2(KSW) + PRESG2(KSE) )
           PRESSE = 0.5*( PRESG2(KSE) + PRESG2(KNE) )
           PRESSN = 0.5*( PRESG2(KNW) + PRESG2(KNE) )
           PRESSW = 0.5*( PRESG2(KSW) + PRESG2(KNW) )

CVD$       NOLSTVAL
           DO 20 IQ = 1, NEQNFL
              DPENFA(IQ,1) = 0.5*( DPENG2(IQ,KSW) + DPENG2(IQ,KSE) )
              DPENFA(IQ,2) = 0.5*( DPENG2(IQ,KSE) + DPENG2(IQ,KNE) )
              DPENFA(IQ,3) = 0.5*( DPENG2(IQ,KNE) + DPENG2(IQ,KNW) )
              DPENFA(IQ,4) = 0.5*( DPENG2(IQ,KNW) + DPENG2(IQ,KSW) )
20         CONTINUE

           UCOMPS = DPENFA(2,1)/DPENFA(1,1)
```

```
      VCOMPS = DPENFA(3,1)/DPENFA(1,1)
      UCOMPE = DPENFA(2,2)/DPENFA(1,2)
      VCOMPE = DPENFA(3,2)/DPENFA(1,2)
      UCOMPN = DPENFA(2,3)/DPENFA(1,3)
      VCOMPN = DPENFA(3,3)/DPENFA(1,3)
      UCOMPW = DPENFA(2,4)/DPENFA(1,4)
      VCOMPW = DPENFA(3,4)/DPENFA(1,4)

C     ----------
C     FLUX TERMS
C     ----------
C     SOUTH

      BIGFS(1) = DPENFA(2,1)
      BIGFS(2) = DPENFA(2,1)*UCOMPS + PRESSS
      BIGFS(3) = DPENFA(2,1)*VCOMPS
      BIGFS(4) = UCOMPS*(DPENFA(4,1) + PRESSS)

      BIGGS(1) = DPENFA(3,1)
      BIGGS(2) = BIGFS(3)
      BIGGS(3) = DPENFA(3,1)*VCOMPS + PRESSS
      BIGGS(4) = VCOMPS*(DPENFA(4,1) + PRESSS)

C     EAST

      BIGFE(1) = DPENFA(2,2)
      BIGFE(2) = DPENFA(2,2)*UCOMPE + PRESSE
      BIGFE(3) = DPENFA(2,2)*VCOMPE
      BIGFE(4) = UCOMPE*(DPENFA(4,2) + PRESSE)
      BIGGE(1) = DPENFA(3,2)
      BIGGE(2) = BIGFE(3)
      BIGGE(3) = DPENFA(3,2)*VCOMPE + PRESSE
      BIGGE(4) = VCOMPE*(DPENFA(4,2) + PRESSE)

C     NORTH

      BIGFN(1) = DPENFA(2,3)
      BIGFN(2) = DPENFA(2,3)*UCOMPN + PRESSN
      BIGFN(3) = DPENFA(2,3)*VCOMPN
      BIGFN(4) = UCOMPN*(DPENFA(4,3) + PRESSN)
      BIGGN(1) = DPENFA(3,3)
      BIGGN(2) = BIGFN(3)
      BIGGN(3) = DPENFA(3,3)*VCOMPN + PRESSN
      BIGGN(4) = VCOMPN*(DPENFA(4,3) + PRESSN)

C     WEST

      BIGFW(1) = DPENFA(2,4)
      BIGFW(2) = DPENFA(2,4)*UCOMPW + PRESSW
      BIGFW(3) = DPENFA(2,4)*VCOMPW
      BIGFW(4) = UCOMPW*(DPENFA(4,4) + PRESSW)
      BIGGW(1) = DPENFA(3,4)
      BIGGW(2) = BIGFW(3)
      BIGGW(3) = DPENFA(3,4)*VCOMPW + PRESSW
      BIGGW(4) = VCOMPW*(DPENFA(4,4) + PRESSW)

C     OTHER FLUX TERMS ASSOCIATED WITH CHEMISTRY
```

```
CVD$      NOLSTVAL
CVD$      NOVECTOR
          DO 30 JS = NEQBAS+1, NEQNFL
             BIGFS(JS) = DPENFA(JS,1)*UCOMPS
             BIGGS(JS) = DPENFA(JS,1)*VCOMPS
             BIGFE(JS) = DPENFA(JS,2)*UCOMPE
             BIGGE(JS) = DPENFA(JS,2)*VCOMPE
             BIGFN(JS) = DPENFA(JS,3)*UCOMPN
             BIGGN(JS) = DPENFA(JS,3)*VCOMPN
             BIGFW(JS) = DPENFA(JS,4)*UCOMPW
             BIGGW(JS) = DPENFA(JS,4)*VCOMPW
30        CONTINUE

C         --------------
C         JACOBIAN TERMS
C         --------------
C
C         DEFINE DEPENDENT VARIABLES AT THE CENTER OF THE CELL

CVD$      NOLSTVAL
          DO 40 IQ = 1, NEQNFL
             DPENJA(IQ) = 0.25*( DPENFA(IQ,1) + DPENFA(IQ,2) +
     1                           DPENFA(IQ,3) + DPENFA(IQ,4) )
40        CONTINUE

C         SET UP THE QUANTITIES NEEDED TO COMPUTE SOURCE TERMS AND
C         JACOBIANS

CVD$      NOLSTVAL
CVD$      NOVECTOR
          DO 50 IQ = NEQBAS+1, NEQNFL
             DELTA     = 0.001*DPENJA(IQ)
             IF (DELTA .EQ. 0.) DELTA = 0.001
             UTOP(IQ)  = DPENJA(IQ) + DELTA
             TOTAL(IQ) = DELTA
50        CONTINUE
C
C         NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
C         UCOMPR, VCOMPR, GAMAPR, YSPEPR ETC. AND GET THE SOURCE TERMS
C         FOR THE CELL
C
          SONDPR = CELLTI(ICELL)
          CALL FRSOUR
C
CVD$      NOLSTVAL
CVD$      NOVECTOR
          DO 60 JS = NEQBAS+1, NEQNFL
             BWCELL(JS) = BIGWJA(JS)
60        CONTINUE

          UCOMPC = UCOMPR
          VCOMPC = VCOMPR
          U2     = UCOMPR*UCOMPR
          V2     = VCOMPR*VCOMPR

          GM1    = GAMAPR - 1.
```

```
      GM3    = GM1 - 2.
      PAEBR  = (BEPSPR+PRESPR)/RHORPR

      FUJACO(1,2) = 1.
      GUJACO(1,3) = 1.
C
      GUJACO(2,1) = -UCOMPC*VCOMPC
      GUJACO(2,2) =  VCOMPC
      GUJACO(2,3) =  UCOMPC
C
      FUJACO(3,1) = GUJACO(2,1)
      FUJACO(3,2) = GUJACO(2,2)
      FUJACO(3,3) = GUJACO(2,3)
C
      FUJACO(2,1) = 0.5*(GM3*U2 + GM1*V2)
      FUJACO(2,2) = -GM3*UCOMPR
      FUJACO(2,3) = -GM1*VCOMPR
      FUJACO(2,4) =  GM1
C
      GUJACO(3,1) = FUJACO(2,1) - V2 + U2
      GUJACO(3,2) = FUJACO(2,2) - 2.*UCOMPC
      GUJACO(3,3) = FUJACO(2,3) - 2.*VCOMPC
      GUJACO(3,4) = FUJACO(2,4)
C
      FUJACO(4,1) = UCOMPR*(FUJACO(2,1) + U2 - PAEBR)
      FUJACO(4,2) = PAEBR - 2.*U2 + UCOMPR*FUJACO(2,2)
      FUJACO(4,3) = UCOMPR*FUJACO(2,3)
      FUJACO(4,4) = UCOMPR*(FUJACO(2,4) + 1.)
C
      GUJACO(4,1) = VCOMPR*(FUJACO(2,1) + U2 - PAEBR)
      GUJACO(4,2) = VCOMPR*(FUJACO(2,2) - 2.*UCOMPR)
      GUJACO(4,3) = VCOMPR*FUJACO(2,3) + PAEBR
      GUJACO(4,4) = VCOMPR*(FUJACO(2,4) + 1.)
C
CVD$  NOLSTVAL
CVD$  NOVECTOR
CVD$  NODEPCHK
      DO 70 JS = NEQBAS + 1, NEQNFL

         YS            = DPENJA(JS)/DPENJA(1)

         FUJACO(JS,1 ) = -UCOMPC*YS
         FUJACO(JS,2 ) =  YS
         FUJACO(JS,JS) =  UCOMPC

         GUJACO(JS,1 ) = -VCOMPC*YS
         GUJACO(JS,3 ) =  YS
         GUJACO(JS,JS) =  VCOMPC

70    CONTINUE

      F2BOT = DPENJA(2)*UCOMPC + PRESPR

CVD$  NOLSTVAL
CVD$  NOVECTOR
      DO 80 JS   = NEQBAS + 1, NEQNFL
         WBOT(JS) = BIGWJA(JS)
```

```
80          CONTINUE

C           COMPUTE THE NUMERICAL JACOBIANS BY TAKING FORWARD DIFFERNCES

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 110 LS = NEQBAS+1, NEQNFL

C              COMPUTE VALUES AT TOP
               UDUMMY     = DPENJA(LS)
               DPENJA(LS) = UTOP(LS)
               CALL         E2SOUR
               F2TOP      = BGF2JA

CVD$           NOLSTVAL
CVD$           NOVECTOR
               DO 90 JS   = NEQBAS + 1, NEQNFL
                  WTOP(JS) = BIGWJA(JS)
90             CONTINUE

C              RESET THE VALUE OF THE DEPENDENT VARIABLE

               DPENJA(LS) = UDUMMY

C              NOW TAKE FORWARD DIFFERENCES

               FUJACO(2,LS) = (F2TOP - F2BOT)/TOTAL(LS)
               GUJACO(3,LS) = FUJACO(2,LS)
               FUJACO(4,LS) = FUJACO(2,LS)*UCOMPC
               GUJACO(4,LS) = FUJACO(2,LS)*VCOMPC

CVD$           NOLSTVAL
CVD$           NOVECTOR
               DO 100 JS = NEQBAS + 1, NEQNFL
                  WUJACO(JS,LS) = (WTOP(JS) - WBOT(JS))/TOTAL(LS)
100            CONTINUE

110         CONTINUE

C           ------------------------------
C           FIRST ORDER CELL CHANGE DUCELL
C           ------------------------------

C           CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE

CVD$        NOLSTVAL
            DO 120 J = 1, NEQNFL
               DUCELL(J) = BWCELL(J)*CELLTI(ICELL) + DTDVOL*(
     1              BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1              BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1              BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
     1              BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )
120         CONTINUE

C           --------------------
C           JACOBIAN CHANGE BLOCK
C           --------------------
```

```
C          COMPUTE CHANGES DUE TO JACOBIANS

CVD$       NOLSTVAL
           DO 140 J = 1, NEQNFL
              DFCELL = 0.
              DGCELL = 0.
              DWCELL = 0.

              DO 130 K = 1, NEQNFL
                 DFCELL = DFCELL + FUJACO(J,K)*DUCELL(K)
                 DGCELL = DGCELL + GUJACO(J,K)*DUCELL(K)
                 DWCELL = DWCELL + WUJACO(J,K)*DUCELL(K)
130           CONTINUE

C             TRANSFORM THE JACOBIAN CHANGES (ONLY FU AND GU) AND
C             MULTIPLY WITH THEIR RESPECTIVE SCALINGS OF TIME

              TEMPF  = DFCELL
              DFCELL = DTDVOL*( TEMPF*DYNSM2(ICELL)
     1                         -DGCELL*DXNSM2(ICELL))
              DGCELL = DTDVOL*(-TEMPF*DYEWM2(ICELL)
     1                         +DGCELL*DXEWM2(ICELL))
              DWCELL = 0.5*CELLTI(ICELL)*DWCELL

C             ----------------
C             DIFFUSION  TERMS
C             ----------------

C             COMPUTE THE DIFFUSION TERMS FOR THE FOUR EDGES

              SIGGSW = SIGGE2(KSW)*DPENG2(J,KSW)
              SIGGSE = SIGGE2(KSE)*DPENG2(J,KSE)
              SIGGNE = SIGGE2(KNE)*DPENG2(J,KNE)
              SIGGNW = SIGGE2(KNW)*DPENG2(J,KNW)

C             COMPUTE THE DIFFUSION TERM FOR THE WHOLE CELL

              SIGCEL= 0.25*(SIGGSW + SIGGSE + SIGGNE + SIGGNW)
              SIGGSW = SIGCEL - SIGGSW
              SIGGSE = SIGCEL - SIGGSE
              SIGGNE = SIGCEL - SIGGNE
              SIGGNW = SIGCEL - SIGGNW
C
              SIGGSW = DSDIFF*SIGGSW
              SIGGSE = DSDIFF*SIGGSE
              SIGGNE = DSDIFF*SIGGNE
              SIGGNW = DSDIFF*SIGGNW

C             ----------------------
C             COMPUTATION OF CHANGES
C             ----------------------

C             FOCIT IS DUCELL; FIND SOCIT AND CORNER CHANGES

              SOCITSW  = - DFCELL - DGCELL + DWCELL
              SOCITNW  = - DFCELL + DGCELL + DWCELL
```

```
            SOCITNE  = + DFCELL + DGCELL + DWCELL
            SOCITSE  = + DFCELL - DGCELL + DWCELL
          -
            DELSW(J) = 0.25*( DUCELL(J) + SOCITSW + SIGGSW )
            DELNW(J) = 0.25*( DUCELL(J) + SOCITNW + SIGGNW )
            DELNE(J) = 0.25*( DUCELL(J) + SOCITNE + SIGGNE )
            DELSE(J) = 0.25*( DUCELL(J) + SOCITSE + SIGGSE )

140         CONTINUE

C           ----------------------
C           DISTRIBUTION OF CHANGES
C           ----------------------

C           DISTRIBUTE CONVECTIVE AND DIFFUSIVE CHANGES

CVD$        NOLSTVAL
            DO 150 J = 1, NEQNFL
               CHNGE2(J,KSW) = CHNGE2(J,KSW) + DELSW(J)
               CHNGE2(J,KNW) = CHNGE2(J,KNW) + DELNW(J)
               CHNGE2(J,KSE) = CHNGE2(J,KSE) + DELSE(J)
               CHNGE2(J,KNE) = CHNGE2(J,KNE) + DELNE(J)
150         CONTINUE

160      CONTINUE
C
         RETURN

C        USE IMPLICIT SOURCE TERMS
C        STEP THROUGH EACH CELL AT THIS LEVEL
C

CVD$     NOLSTVAL
310      DO 560 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C           ----------------------
C           CELL/NODE DETERMINATION
C           ----------------------

C           FIND THE CELL TO BE INTEGRATED

            ICELL = ICELTI(JCELL)

C           SET UP NODE POINTERS FOR THIS CELL

            KSW = ICELG2( 2,ICELL)
            KSE = ICELG2( 4,ICELL)
            KNE = ICELG2( 6,ICELL)
            KNW = ICELG2( 8,ICELL)

C           --------
C           GEOMETRY
C           --------
C
C           GEOMETRY OF ALL CELL CORNERS

            XSW = GEOMG2(1,KSW)
```

```
      YSW = GEOMG2(2,KSW)

      XSE = GEOMG2(1,KSE)
      YSE = GEOMG2(2,KSE)

      XNE = GEOMG2(1,KNE)
      YNE = GEOMG2(2,KNE)

      XNW = GEOMG2(1,KNW)
      YNW = GEOMG2(2,KNW)
C
C     THE RATIO DELTA-t TO CELL VOLUME
      DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

C     COMPUTE THE AVERAGE COEFFICIENT FOR DIFFUSION

      DSDIFF = 0.5*PERIM2(ICELL)*DTDVOL

C     -------------
C     FACIAL VALUES
C     -------------
C
C     COMPUTE THE DEPENDENT VARIABLES AT THE FACES

      PRESSS = 0.5*( PRESG2(KSW) + PRESG2(KSE) )
      PRESSE = 0.5*( PRESG2(KSE) + PRESG2(KNE) )
      PRESSN = 0.5*( PRESG2(KNW) + PRESG2(KNE) )
      PRESSW = 0.5*( PRESG2(KSW) + PRESG2(KNW) )

CVD$  NOLSTVAL
      DO 320 IQ = 1, NEQNFL
         DPENFA(IQ,1) = 0.5*( DPENG2(IQ,KSW) + DPENG2(IQ,KSE) )
         DPENFA(IQ,2) = 0.5*( DPENG2(IQ,KSE) + DPENG2(IQ,KNE) )
         DPENFA(IQ,3) = 0.5*( DPENG2(IQ,KNE) + DPENG2(IQ,KNW) )
         DPENFA(IQ,4) = 0.5*( DPENG2(IQ,KNW) + DPENG2(IQ,KSW) )
320   CONTINUE

      UCOMPS = DPENFA(2,1)/DPENFA(1,1)
      VCOMPS = DPENFA(3,1)/DPENFA(1,1)
      UCOMPE = DPENFA(2,2)/DPENFA(1,2)
      VCOMPE = DPENFA(3,2)/DPENFA(1,2)
      UCOMPN = DPENFA(2,3)/DPENFA(1,3)
      VCOMPN = DPENFA(3,3)/DPENFA(1,3)
      UCOMPW = DPENFA(2,4)/DPENFA(1,4)
      VCOMPW = DPENFA(3,4)/DPENFA(1,4)

C     ----------
C     FLUX TERMS
C     ----------
C     SOUTH

      BIGFS(1) = DPENFA(2,1)
      BIGFS(2) = DPENFA(2,1)*UCOMPS + PRESSS
      BIGFS(3) = DPENFA(2,1)*VCOMPS
      BIGFS(4) = UCOMPS*(DPENFA(4,1) + PRESSS)

      BIGGS(1) = DPENFA(3,1)
```

```
            BIGGS(2) = BIGFS(3)
            BIGGS(3) = DPENFA(3,1)*VCOMPS + PRESSS
            BIGGS(4) = VCOMPS*(DPENFA(4,1) + PRESSS)

C           EAST

            BIGFE(1) = DPENFA(2,2)
            BIGFE(2) = DPENFA(2,2)*UCOMPE + PRESSE
            BIGFE(3) = DPENFA(2,2)*VCOMPE
            BIGFE(4) = UCOMPE*(DPENFA(4,2) + PRESSE)
            BIGGE(1) = DPENFA(3,2)
            BIGGE(2) = BIGFE(3)
            BIGGE(3) = DPENFA(3,2)*VCOMPE + PRESSE
            BIGGE(4) = VCOMPE*(DPENFA(4,2) + PRESSE)

C           NORTH

            BIGFN(1) = DPENFA(2,3)
            BIGFN(2) = DPENFA(2,3)*UCOMPN + PRESSN
            BIGFN(3) = DPENFA(2,3)*VCOMPN
            BIGFN(4) = UCOMPN*(DPENFA(4,3) + PRESSN)
            BIGGN(1) = DPENFA(3,3)
            BIGGN(2) = BIGFN(3)
            BIGGN(3) = DPENFA(3,3)*VCOMPN + PRESSN
            BIGGN(4) = VCOMPN*(DPENFA(4,3) + PRESSN)

C           WEST

            BIGFW(1) = DPENFA(2,4)
            BIGFW(2) = DPENFA(2,4)*UCOMPW + PRESSW
            BIGFW(3) = DPENFA(2,4)*VCOMPW
            BIGFW(4) = UCOMPW*(DPENFA(4,4) + PRESSW)
            BIGGW(1) = DPENFA(3,4)
            BIGGW(2) = BIGFW(3)
            BIGGW(3) = DPENFA(3,4)*VCOMPW + PRESSW
            BIGGW(4) = VCOMPW*(DPENFA(4,4) + PRESSW)

C           OTHER FLUX TERMS ASSOCIATED WITH CHEMISTRY

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 330 JS = NEQBAS+1, NEQNFL
               BIGFS(JS) = DPENFA(JS,1)*UCOMPS
               BIGGS(JS) = DPENFA(JS,1)*VCOMPS
               BIGFE(JS) = DPENFA(JS,2)*UCOMPE
               BIGGE(JS) = DPENFA(JS,2)*VCOMPE
               BIGFN(JS) = DPENFA(JS,3)*UCOMPN
               BIGGN(JS) = DPENFA(JS,3)*VCOMPN
               BIGFW(JS) = DPENFA(JS,4)*UCOMPW
               BIGGW(JS) = DPENFA(JS,4)*VCOMPW
330         CONTINUE

C           --------------
C           JACOBIAN TERMS
C           --------------
C
C           DEFINE DEPENDENT VARIABLES AT THE CENTER OF THE CELL
```

```
CVD$      NOLSTVAL
          DO 340 IQ = 1, NEQNFL
             DPENJA(IQ) = 0.25*( DPENFA(IQ,1) + DPENFA(IQ,2) +
     1                           DPENFA(IQ,3) + DPENFA(IQ,4) )
340       CONTINUE

C         SET UP THE QUANTITIES NEEDED TO COMPUTE SOURCE TERMS AND
C         JACOBIANS

CVD$      NOLSTVAL
CVD$      NOVECTOR
          DO 350 IQ = NEQBAS+1, NEQNFL
             DELTA     = 0.001*DPENJA(IQ)
             IF (DELTA .EQ. 0.) DELTA = 0.001
             UTOP(IQ)  = DPENJA(IQ) + DELTA
             TOTAL(IQ) = DELTA
350       CONTINUE
C
C         NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
C         UCOMPR, VCOMPR, GAMAPR, YSPEPR ETC. AND GET THE SOURCE TERMS
C         FOR THE CELL
C
          SONDPR = CELLTI(ICELL)
          CALL FRSOUR
C
CVD$      NOLSTVAL
CVD$      NOVECTOR
          DO 360 JS = NEQBAS+1, NEQNFL
             BWCELL(JS) = BIGWJA(JS)
360       CONTINUE

          UCOMPC = UCOMPR
          VCOMPC = VCOMPR
          U2     = UCOMPR*UCOMPR
          V2     = VCOMPR*VCOMPR

          GM1    = GAMAPR - 1.
          GM3    = GM1 - 2.
          PAEBR  = (BEPSPR+PRESPR)/RHORPR

          FUJACO(1,2) = 1.
          GUJACO(1,3) = 1.
C
          GUJACO(2,1) = -UCOMPC*VCOMPC
          GUJACO(2,2) =  VCOMPC
          GUJACO(2,3) =  UCOMPC
C
          FUJACO(3,1) = GUJACO(2,1)
          FUJACO(3,2) = GUJACO(2,2)
          FUJACO(3,3) = GUJACO(2,3)
C
          FUJACO(2,1) = 0.5*(GM3*U2 + GM1*V2)
          FUJACO(2,2) = -GM3*UCOMPR
          FUJACO(2,3) = -GM1*VCOMPR
          FUJACO(2,4) =  GM1
C
```

```
            GUJACO(3,1) = FUJACO(2,1) - V2 + U2
            GUJACO(3,2) = FUJACO(2,2) - 2.*UCOMPC
            GUJACO(3,3) = FUJACO(2,3) - 2.*VCOMPC
            GUJACO(3,4) = FUJACO(2,4)
C
            FUJACO(4,1) = UCOMPR*(FUJACO(2,1) + U2 - PAEBR)
            FUJACO(4,2) = PAEBR - 2.*U2 + UCOMPR*FUJACO(2,2)
            FUJACO(4,3) = UCOMPR*FUJACO(2,3)
            FUJACO(4,4) = UCOMPR*(FUJACO(2,4) + 1.)
C
            GUJACO(4,1) = VCOMPR*(FUJACO(2,1) + U2 - PAEBR)
            GUJACO(4,2) = VCOMPR*(FUJACO(2,2) - 2.*UCOMPR)
            GUJACO(4,3) = VCOMPR*FUJACO(2,3) + PAEBR
            GUJACO(4,4) = VCOMPR*(FUJACO(2,4) + 1.)
C
CVD$        NOLSTVAL
CVD$        NOVECTOR
CVD$        NODEPCHK
            DO 370 JS = NEQBAS + 1, NEQNFL

               YS            = DPENJA(JS)/DPENJA(1)

               FUJACO(JS,1 ) = -UCOMPC*YS
               FUJACO(JS,2 ) =  YS
               FUJACO(JS,JS) =  UCOMPC

               GUJACO(JS,1 ) = -VCOMPC*YS
               GUJACO(JS,3 ) =  YS
               GUJACO(JS,JS) =  VCOMPC

370         CONTINUE

            F2BOT = DPENJA(2)*UCOMPC + PRESPR

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 380 JS   = NEQBAS + 1, NEQNFL
               WBOT(JS) = BIGWJA(JS)
380         CONTINUE

C           COMPUTE THE NUMERICAL JACOBIANS BY TAKING FORWARD DIFFERNCES

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 410 LS = NEQBAS+1, NEQNFL

C              COMPUTE VALUES AT TOP
               UDUMMY     = DPENJA(LS)
               DPENJA(LS) = UTOP(LS)
               CALL         E2SOUR
               F2TOP      = BGF2JA

CVD$           NOLSTVAL
CVD$           NOVECTOR
               DO 390 JS   = NEQBAS + 1, NEQNFL
                  WTOP(JS) = BIGWJA(JS)
390            CONTINUE
```

```
C              RESET THE VALUE OF THE DEPENDENT VARIABLE

               DPENJA(LS) = UDUMMY

C              NOW TAKE FORWARD DIFFERENCES

               FUJACO(2,LS) = (F2TOP - F2BOT)/TOTAL(LS)
               GUJACO(3,LS) = FUJACO(2,LS)
               FUJACO(4,LS) = FUJACO(2,LS)*UCOMPC
               GUJACO(4,LS) = FUJACO(2,LS)*VCOMPC

CVD$           NOLSTVAL
CVD$           NOVECTOR
               DO 400 JS = NEQBAS + 1, NEQNFL
                  if (abs(total(ls)) .gt. 1.e-20) then
                  WUJACO(JS,LS) = (WTOP(JS) - WBOT(JS))/TOTAL(LS)
                  else
                  WUJACO(JS,LS) = 0.
                  endif
400            CONTINUE

410         CONTINUE

C           ------------------------------
C           FIRST ORDER CELL CHANGE DUCELL
C           ------------------------------

C           CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE

CVD$        NOLSTVAL
            DO 420 J = 1, NEQNFL
               DUCELL(J) = BWCELL(J)*CELLTI(ICELL) + DTDVOL*(
     1              BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1              BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1              BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
     1              BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )
420         CONTINUE

C           --------------------
C           JACOBIAN CHANGE BLOCK
C           --------------------

C           COMPUTE CHANGES DUE TO JACOBIANS

CVD$        NOLSTVAL
            DO 440 J = 1, NEQNFL
               DFCELL = 0.
               DGCELL = 0.

               DO 430 K = 1, NEQNFL
                  DFCELL = DFCELL + FUJACO(J,K)*DUCELL(K)
                  DGCELL = DGCELL + GUJACO(J,K)*DUCELL(K)
430            CONTINUE

C              TRANSFORM THE JACOBIAN CHANGES (ONLY FU AND GU) AND
C              MULTIPLY WITH THEIR RESPECTIVE SCALINGS OF TIME
```

```
              TEMPF  = DFCELL
             ~DFCELL = DTDVOL*( TEMPF*DYNSM2(ICELL)
     1                         -DGCELL*DXNSM2(ICELL))
              DGCELL = DTDVOL*(-TEMPF*DYEWM2(ICELL)
     1                         +DGCELL*DXEWM2(ICELL))

C             ---------------
C             DIFFUSION  TERMS
C             ---------------

C             COMPUTE THE DIFFUSION TERMS FOR THE FOUR EDGES

              SIGGSW = SIGGE2(KSW)*DPENG2(J,KSW)
              SIGGSE = SIGGE2(KSE)*DPENG2(J,KSE)
              SIGGNE = SIGGE2(KNE)*DPENG2(J,KNE)
              SIGGNW = SIGGE2(KNW)*DPENG2(J,KNW)

C             COMPUTE THE DIFFUSION TERM FOR THE WHOLE CELL

              SIGCEL= 0.25*(SIGGSW + SIGGSE + SIGGNE + SIGGNW)
              SIGGSW = SIGCEL - SIGGSW
              SIGGSE = SIGCEL - SIGGSE
              SIGGNE = SIGCEL - SIGGNE
              SIGGNW = SIGCEL - SIGGNW
C
              SIGGSW = DSDIFF*SIGGSW
              SIGGSE = DSDIFF*SIGGSE
              SIGGNE = DSDIFF*SIGGNE
              SIGGNW = DSDIFF*SIGGNW

C             ----------------------
C             COMPUTATION OF CHANGES
C             ----------------------

C             FOCIT IS DUCELL; FIND SOCIT AND CORNER CHANGES

              SOCITSW  = - DFCELL - DGCELL
              SOCITNW  = - DFCELL + DGCELL
              SOCITNE  = + DFCELL + DGCELL
              SOCITSE  = + DFCELL - DGCELL

              DELSW(J) = 0.25*( DUCELL(J) + SOCITSW + SIGGSW )
              DELNW(J) = 0.25*( DUCELL(J) + SOCITNW + SIGGNW )
              DELNE(J) = 0.25*( DUCELL(J) + SOCITNE + SIGGNE )
              DELSE(J) = 0.25*( DUCELL(J) + SOCITSE + SIGGSE )

440           CONTINUE
C
C             DO IMPLICIT SOURCE TERMS

              CALL PTIMP2 (KSW, ICELL, DELSW)
              CALL PTIMP2 (KSE, ICELL, DELSE)
              CALL PTIMP2 (KNW, ICELL, DELNW)
              CALL PTIMP2 (KNE, ICELL, DELNE)

C             ----------------------
```

```
C          DISTRIBUTION OF CHANGES
C          ----------------------
C          DISTRIBUTE CONVECTIVE AND DIFFUSIVE CHANGES

CVD$       NOLSTVAL
           DO 450 J = 1, NEQNFL
              CHNGE2(J,KSW) = CHNGE2(J,KSW) + DELSW(J)
              CHNGE2(J,KNW) = CHNGE2(J,KNW) + DELNW(J)
              CHNGE2(J,KSE) = CHNGE2(J,KSE) + DELSE(J)
              CHNGE2(J,KNE) = CHNGE2(J,KNE) + DELNE(J)
450        CONTINUE

560     CONTINUE
C
        RETURN
C
C       USE EXPLICIT SOURCE TERMS, KEEPING THE CHEMISTRY FROZEN
C       STEP THROUGH EACH CELL AT THIS LEVEL
C
CVD$    NOLSTVAL
610     DO 710 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C          ----------------------
C          CELL/NODE DETERMINATION
C          ----------------------

C          FIND THE CELL TO BE INTEGRATED

           ICELL = ICELTI(JCELL)

C          SET UP NODE POINTERS FOR THIS CELL

           KSW = ICELG2( 2,ICELL)
           KSE = ICELG2( 4,ICELL)
           KNE = ICELG2( 6,ICELL)
           KNW = ICELG2( 8,ICELL)

C          --------
C          GEOMETRY
C          --------
C
C          GEOMETRY OF ALL CELL CORNERS

           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)

           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)

           XNE = GEOMG2(1,KNE)
           YNE = GEOMG2(2,KNE)

           XNW = GEOMG2(1,KNW)
           YNW = GEOMG2(2,KNW)
C
C          THE RATIO DELTA-t TO CELL VOLUME
```

```
            DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

C           COMPUTE THE AVERAGE COEFFICIENT FOR DIFFUSION

            DSDIFF = 0.5*PERIM2(ICELL)*DTDVOL

C           -------------
C           FACIAL VALUES
C           -------------
C
C           COMPUTE THE DEPENDENT VARIABLES AT THE FACES

            PRESSS = 0.5*( PRESG2(KSW) + PRESG2(KSE) )
            PRESSE = 0.5*( PRESG2(KSE) + PRESG2(KNE) )
            PRESSN = 0.5*( PRESG2(KNW) + PRESG2(KNE) )
            PRESSW = 0.5*( PRESG2(KSW) + PRESG2(KNW) )

CVD$        NOLSTVAL
            DO 620 IQ = 1, 4
               DPENFA(IQ,1) = 0.5*( DPENG2(IQ,KSW) + DPENG2(IQ,KSE) )
               DPENFA(IQ,2) = 0.5*( DPENG2(IQ,KSE) + DPENG2(IQ,KNE) )
               DPENFA(IQ,3) = 0.5*( DPENG2(IQ,KNE) + DPENG2(IQ,KNW) )
               DPENFA(IQ,4) = 0.5*( DPENG2(IQ,KNW) + DPENG2(IQ,KSW) )
620         CONTINUE

            UCOMPS = DPENFA(2,1)/DPENFA(1,1)
            VCOMPS = DPENFA(3,1)/DPENFA(1,1)
            UCOMPE = DPENFA(2,2)/DPENFA(1,2)
            VCOMPE = DPENFA(3,2)/DPENFA(1,2)
            UCOMPN = DPENFA(2,3)/DPENFA(1,3)
            VCOMPN = DPENFA(3,3)/DPENFA(1,3)
            UCOMPW = DPENFA(2,4)/DPENFA(1,4)
            VCOMPW = DPENFA(3,4)/DPENFA(1,4)

C           ----------
C           FLUX TERMS
C           ----------
C           SOUTH

            BIGFS(1) = DPENFA(2,1)
            BIGFS(2) = DPENFA(2,1)*UCOMPS + PRESSS
            BIGFS(3) = DPENFA(2,1)*VCOMPS
            BIGFS(4) = UCOMPS*(DPENFA(4,1) + PRESSS)

            BIGGS(1) = DPENFA(3,1)
            BIGGS(2) = BIGFS(3)
            BIGGS(3) = DPENFA(3,1)*VCOMPS + PRESSS
            BIGGS(4) = VCOMPS*(DPENFA(4,1) + PRESSS)

C           EAST

            BIGFE(1) = DPENFA(2,2)
            BIGFE(2) = DPENFA(2,2)*UCOMPE + PRESSE
            BIGFE(3) = DPENFA(2,2)*VCOMPE
            BIGFE(4) = UCOMPE*(DPENFA(4,2) + PRESSE)
            BIGGE(1) = DPENFA(3,2)
            BIGGE(2) = BIGFE(3)
```

```
      BIGGE(3) = DPENFA(3,2)*VCOMPE + PRESSE
      BIGGE(4) = VCOMPE*(DPENFA(4,2) + PRESSE)
       -
C     NORTH

      BIGFN(1) = DPENFA(2,3)
      BIGFN(2) = DPENFA(2,3)*UCOMPN + PRESSN
      BIGFN(3) = DPENFA(2,3)*VCOMPN
      BIGFN(4) = UCOMPN*(DPENFA(4,3) + PRESSN)
      BIGGN(1) = DPENFA(3,3)
      BIGGN(2) = BIGFN(3)
      BIGGN(3) = DPENFA(3,3)*VCOMPN + PRESSN
      BIGGN(4) = VCOMPN*(DPENFA(4,3) + PRESSN)

C     WEST

      BIGFW(1) = DPENFA(2,4)
      BIGFW(2) = DPENFA(2,4)*UCOMPW + PRESSW
      BIGFW(3) = DPENFA(2,4)*VCOMPW
      BIGFW(4) = UCOMPW*(DPENFA(4,4) + PRESSW)
      BIGGW(1) = DPENFA(3,4)
      BIGGW(2) = BIGFW(3)
      BIGGW(3) = DPENFA(3,4)*VCOMPW + PRESSW
      BIGGW(4) = VCOMPW*(DPENFA(4,4) + PRESSW)

C     --------------
C     JACOBIAN TERMS
C     --------------
C
C     DEFINE DEPENDENT VARIABLES AT THE CENTER OF THE CELL

CVD$  NOLSTVAL
      DO 630 IQ = 1, 4
         DPENJA(IQ) = 0.25*( DPENFA(IQ,1) + DPENFA(IQ,2) +
     1                       DPENFA(IQ,3) + DPENFA(IQ,4) )
630   CONTINUE
C
C     NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
C     UCOMPC, VCOMPC, GAMAPR, YSPEPR ETC. AND GET THE SOURCE TERMS
C     FOR THE CELL
C
      UCOMPC = DPENJA(2)/DPENJA(1)
      VCOMPC = DPENJA(3)/DPENJA(1)
      U2     = UCOMPC*UCOMPC
      V2     = VCOMPC*VCOMPC
      BEPSPR = DPENJA(4)
      BEU    = BEPSPR/DPENJA(1)
      VELO2U = U2 + V2
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BEU
      VELO2  = FMREFL*VELO2U

C     COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

      SUMY = 0.
```

```
      DO 640 IS = 1, NEQSCH
         JS         = NEQBAS + IS
       - YSPEPR(IS) = DPENJA(JS)/DPENJA(1)
         SUMY       = SUMY + YSPEPR(IS)
640   CONTINUE

      YNEXT            = 1. - SUMY - YNRTCH
      IF (YNEXT .LT. 0.) YNEXT = 0.
      YSPEPR(NEQSCH+1) = YNEXT
C
      SYSHFS = 0.
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
C     COMPUTE THE TEMPERATURE IN DEGREE K AND ALSO
C
      DO 650 IS = 1, NSPECH
         SYSHFS     = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
         SYSCPS     = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
         SYSBMS     = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
         BIGAM      = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
650   CONTINUE
C
C     COMPUTE TEMPERATURE IN DEGREE K AND SOME RELATED QUANTITIES

      BIGBM   = SYSCPS - UGASFL*SYSBMS
      BIGCM   = BE - 0.5*VELO2 - SYSHFS + TREFCH*SYSCPS
     1                  + 0.5*TREFCH*TREFCH*BIGAM
      IF (BIGAM .LT. 1.E-10) THEN
         TEMP   = BIGCM/BIGBM
      ELSE
         DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
         TEMP   = ( SQRT(DISCRI)-BIGBM )/BIGAM
      ENDIF

      BIGAMT = BIGAM *TEMP
      SYSCVS = BIGBM + BIGAMT
      GAMAPR = (SYSCPS+BIGAMT)/SYSCVS
C
C     NORMALIZE THE TEMPERATURE
C
      TEMPU = TEMP/TREFFL
C
C     COMPUTE THE DIMENSIONLESS PRESSURE
C
      PRESPR = DPENJA(1)*TEMPU*AMWTFL*SYSBMS

      GM1    = GAMAPR - 1.
      GM3    = GM1 - 2.
      PAEBR  = (DPENJA(4)+PRESPR)/DPENJA(1)

      FUJACO(1,2) = 1.
      GUJACO(1,3) = 1.
C
      GUJACO(2,1) = -UCOMPC*VCOMPC
      GUJACO(2,2) =  VCOMPC
```

```
            GUJACO(2,3) =  UCOMPC
C
            FUJACO(3,1) = GUJACO(2,1)
            FUJACO(3,2) = GUJACO(2,2)
            FUJACO(3,3) = GUJACO(2,3)
C
            FUJACO(2,1) = 0.5*(GM3*U2 + GM1*V2)
            FUJACO(2,2) = -GM3*UCOMPC
            FUJACO(2,3) = -GM1*VCOMPC
            FUJACO(2,4) =  GM1
C
            GUJACO(3,1) = FUJACO(2,1) - V2 + U2
            GUJACO(3,2) = FUJACO(2,2) - 2.*UCOMPC
            GUJACO(3,3) = FUJACO(2,3) - 2.*VCOMPC
            GUJACO(3,4) = FUJACO(2,4)
C
            FUJACO(4,1) = UCOMPC*(FUJACO(2,1) + U2 - PAEBR)
            FUJACO(4,2) = PAEBR - 2.*U2 + UCOMPC*FUJACO(2,2)
            FUJACO(4,3) = UCOMPC*FUJACO(2,3)
            FUJACO(4,4) = UCOMPC*(FUJACO(2,4) + 1.)
C
            GUJACO(4,1) = VCOMPC*(FUJACO(2,1) + U2 - PAEBR)
            GUJACO(4,2) = VCOMPC*(FUJACO(2,2) - 2.*UCOMPC)
            GUJACO(4,3) = VCOMPC*FUJACO(2,3) + PAEBR
            GUJACO(4,4) = VCOMPC*(FUJACO(2,4) + 1.)
C
C           ------------------------------
C           FIRST ORDER CELL CHANGE DUCELL
C           ------------------------------

C           CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE

CVD$        NOLSTVAL
            DO 660 J = 1, 4
               DUCELL(J) = DTDVOL*(
     1              BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1              BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1              BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
     1              BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )
660         CONTINUE

C           ---------------------
C           JACOBIAN CHANGE BLOCK
C           ---------------------

C           COMPUTE CHANGES DUE TO JACOBIANS

CVD$        NOLSTVAL
            DO 680 J = 1, 4
               DFCELL = 0.
               DGCELL = 0.

               DO 670 K = 1, 4
                  DFCELL = DFCELL + FUJACO(J,K)*DUCELL(K)
                  DGCELL = DGCELL + GUJACO(J,K)*DUCELL(K)
670            CONTINUE
```

```
C             TRANSFORM THE JACOBIAN CHANGES (ONLY FU AND GU) AND
C             MULTIPLY WITH THEIR RESPECTIVE SCALINGS OF TIME

              TEMPF  = DFCELL
              DFCELL = DTDVOL*( TEMPF*DYNSM2(ICELL)
     1                         -DGCELL*DXNSM2(ICELL))
              DGCELL = DTDVOL*(-TEMPF*DYEWM2(ICELL)
     1                         +DGCELL*DXEWM2(ICELL))

C             ----------------
C             DIFFUSION  TERMS
C             ----------------

C             COMPUTE THE DIFFUSION TERMS FOR THE FOUR EDGES

              SIGGSW = SIGGE2(KSW)*DPENG2(J,KSW)
              SIGGSE = SIGGE2(KSE)*DPENG2(J,KSE)
              SIGGNE = SIGGE2(KNE)*DPENG2(J,KNE)
              SIGGNW = SIGGE2(KNW)*DPENG2(J,KNW)

C             COMPUTE THE DIFFUSION TERM FOR THE WHOLE CELL

              SIGCEL= 0.25*(SIGGSW + SIGGSE + SIGGNE + SIGGNW)
              SIGGSW = SIGCEL - SIGGSW
              SIGGSE = SIGCEL - SIGGSE
              SIGGNE = SIGCEL - SIGGNE
              SIGGNW = SIGCEL - SIGGNW
C
              SIGGSW = DSDIFF*SIGGSW
              SIGGSE = DSDIFF*SIGGSE
              SIGGNE = DSDIFF*SIGGNE
              SIGGNW = DSDIFF*SIGGNW

C             ---------------------
C             COMPUTATION OF CHANGES
C             ---------------------

C             FOCIT IS DUCELL; FIND SOCIT AND CORNER CHANGES

              SOCITSW  = - DFCELL - DGCELL
              SOCITNW  = - DFCELL + DGCELL
              SOCITNE  = + DFCELL + DGCELL
              SOCITSE  = + DFCELL - DGCELL

              DELSW(J) = 0.25*( DUCELL(J) + SOCITSW + SIGGSW )
              DELNW(J) = 0.25*( DUCELL(J) + SOCITNW + SIGGNW )
              DELNE(J) = 0.25*( DUCELL(J) + SOCITNE + SIGGNE )
              DELSE(J) = 0.25*( DUCELL(J) + SOCITSE + SIGGSE )

680           CONTINUE

              DO 690 J = 5, NEQNFL
                 DELSW(J) = DELSW(1)*YSPEPR(J-4)
                 DELNW(J) = DELNW(1)*YSPEPR(J-4)
                 DELNE(J) = DELNE(1)*YSPEPR(J-4)
                 DELNW(J) = DELNW(1)*YSPEPR(J-4)
690           CONTINUE
```

```
C          ----------------------
C          DISTRIBUTION OF CHANGES
C          ----------------------

C          DISTRIBUTE CONVECTIVE AND DIFFUSIVE CHANGES

CVD$       NOLSTVAL
           DO 700 J = 1, NEQNFL
              CHNGE2(J,KSW) = CHNGE2(J,KSW) + DELSW(J)
              CHNGE2(J,KNW) = CHNGE2(J,KNW) + DELNW(J)
              CHNGE2(J,KSE) = CHNGE2(J,KSE) + DELSE(J)
              CHNGE2(J,KNE) = CHNGE2(J,KNE) + DELNE(J)
700        CONTINUE

710     CONTINUE
C
        RETURN
        END
```

E2SOL0

```
        SUBROUTINE E2SOL0 (ITGL)

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] E2COMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
        INCLUDE '[.INC] JACOMN.INC/LIST'
        INCLUDE '[.INC] M2COMN.INC/LIST'
        INCLUDE '[.INC] PRCOMN.INC/LIST'
        INCLUDE '[.INC] TICOMN.INC/LIST'
        COMMON/WUCOMN/ WUJACO
        DIMENSION BIGFS (MEQNFL)         ,   BIGFE (MEQNFL)         ,
     1            BIGFN (MEQNFL)         ,   BIGFW (MEQNFL)         ,
     2            BIGGS (MEQNFL)         ,   BIGGE (MEQNFL)         ,
     3            BIGGN (MEQNFL)         ,   BIGGW (MEQNFL)         ,
     4            DELSW (MEQNFL)         ,   DELSE (MEQNFL)         ,
     5            DELNW (MEQNFL)         ,   DELNE (MEQNFL)         ,
     6            BWCELL(MEQNFL)         ,   DPENFA(MEQNFL,2:9)     ,
     7            FUJACO(MEQNFL,MEQNFL) ,   GUJACO(MEQNFL,MEQNFL) ,
     8            WUJACO(MEQNFL,MEQNFL) ,   DUCELL(MEQNFL)

        DIMENSION UTOP  (MEQNFL)         ,   TOTAL (MEQNFL)         ,
     1            WTOP  (MEQNFL)         ,   WBOT  (MEQNFL)

C*********************************************************************

C       THIS SUBROUTINE STEPS THROUGH EACH CELL ON THE SPATIAL LEVEL ITGL
C       AND APPLIES NI'S SCHEME, I.E., INTEGRATES OVER ALL CELLS ON ITGL.
C       IT ALSO COMPUTES THE ANALYTICAL AS WELL NUMERICAL JACOBIANS,
C       BECAUSE THEIR STORAGE IS COSTLY.
C       DPENFA : VALUES OF DEPENDENT VARIABLES AT THE FACES
```

```
C       DPENG2 : VALUES OF DEPENDENT VARIABLES AT THE NODES
C       DPENJA : VALUES OF DEPENDENT VARIABLES FOR COMPUTING JACOBIANS

C***********************************************************************
C
C       INITIALIZE THE JACOBIAN TERMS

        DO 20 JEQ = 1, NEQNFL
           DO 10 IEQ = 1, NEQNFL
              FUJACO(IEQ,JEQ) = 0.
              GUJACO(IEQ,JEQ) = 0.
              WUJACO(IEQ,JEQ) = 0.
10         CONTINUE
20      CONTINUE
C
C       INITIALIZE THE SOURCE TERMS
C
        DO 30 JS = 1, NEQBAS
           BWCELL(JS) = 0.
30      CONTINUE
C
C       STEP THROUGH EACH CELL AT THIS LEVEL
C
        DO 300 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C          ----------------------
C          CELL/NODE DETERMINATION
C          ----------------------

C          FIND THE CELL TO BE INTEGRATED

           ICELL = ICELTI(JCELL)

C          SET UP NODE POINTERS FOR THIS CELL

           KSW = ICELG2( 2,ICELL)
           KS  = ICELG2( 3,ICELL)
           KSE = ICELG2( 4,ICELL)
           KE  = ICELG2( 5,ICELL)
           KNE = ICELG2( 6,ICELL)
           KN  = ICELG2( 7,ICELL)
           KNW = ICELG2( 8,ICELL)
           KW  = ICELG2( 9,ICELL)
           KX  = KAUXG2(   ICELL)

C          --------
C          GEOMETRY
C          --------
C
C          GEOMETRY OF ALL CELL CORNERS

           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)

           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)
```

```
            XNE = GEOMG2(1,KNE)
            YNE = GEOMG2(2,KNE)

            XNW = GEOMG2(1,KNW)
            YNW = GEOMG2(2,KNW)
C
C           THE RATIO DELTA-t TO CELL VOLUME
            DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

C           COMPUTE THE AVERAGE COEFFICIENT FOR DIFFUSION
            DSDIFF = 0.5*PERIM2(ICELL)*DTDVOL

C           -------------
C           FACIAL VALUES
C           -------------
C
C           COMPUTE THE DEPENDENT VARIABLES AT THE FACES
C           FIRST COMPUTE THE VALUES AT THE CORNER NODES
C
            DO 40 IQ = 1, NEQNFL
               DPENFA(IQ,2) = DPENG2(IQ,KSW)
               DPENFA(IQ,4) = DPENG2(IQ,KSE)
               DPENFA(IQ,6) = DPENG2(IQ,KNE)
               DPENFA(IQ,8) = DPENG2(IQ,KNW)
40          CONTINUE

C           VALUES AT SOUTH NODE

            IF (KS .EQ. 0) THEN
               PRESSS = 0.5*(PRESG2(KSW) + PRESG2(KSE))
               DO 50 IQ = 1, NEQNFL
                  DPENFA(IQ,3) = 0.5*(DPENG2(IQ,KSW) + DPENG2(IQ,KSE))
50             CONTINUE
            ELSE
               PRESSS = PRESG2(KS)
               DO 60 IQ = 1, NEQNFL
                  DPENFA(IQ,3) = DPENG2(IQ,KS)
60             CONTINUE
            ENDIF

C           VALUES AT EAST  NODE

            IF (KE .EQ. 0) THEN
               PRESSE = 0.5*(PRESG2(KSE) + PRESG2(KNE))
               DO 70 IQ = 1, NEQNFL
                  DPENFA(IQ,5) = 0.5*(DPENG2(IQ,KSE) + DPENG2(IQ,KNE))
70             CONTINUE
            ELSE
               PRESSE = PRESG2(KE)
               DO 80 IQ = 1, NEQNFL
                  DPENFA(IQ,5) = DPENG2(IQ,KE)
80             CONTINUE
            ENDIF

C           VALUES AT NORTH NODE

            IF (KN .EQ. 0) THEN
```

```
            PRESSN = 0.5*(PRESG2(KNW) + PRESG2(KNE))
            DO 90 IQ = 1, NEQNFL
               DPENFA(IQ,7) = 0.5*(DPENG2(IQ,KNW) + DPENG2(IQ,KNE))
90          CONTINUE
         ELSE
            PRESSN = PRESG2(KN)
            DO 100 IQ = 1, NEQNFL
               DPENFA(IQ,7) = DPENG2(IQ,KN)
100         CONTINUE
         ENDIF

C        VALUES AT WEST  NODE

         IF (KW .EQ. 0) THEN
            PRESSW = 0.5*(PRESG2(KSW) + PRESG2(KNW))
            DO 110 IQ = 1, NEQNFL
               DPENFA(IQ,9) = 0.5*(DPENG2(IQ,KSW) + DPENG2(IQ,KNW))
110         CONTINUE
         ELSE
            PRESSW = PRESG2(KW)
            DO 120 IQ = 1, NEQNFL
               DPENFA(IQ,9) = DPENG2(IQ,KW)
120         CONTINUE
         ENDIF

C        VALUES AT THE WHOLE FACES CAN BE DETERMINED NOW
C        SOUTH FACE

         PRESSS = 0.25*( PRESG2(KSW) + 2.*PRESSS + PRESG2(KSE) )
         DO 130 IQ = 1, NEQNFL
            DPENFA(IQ,3) = 0.25*( DPENFA(IQ,2) + 2.*DPENFA(IQ,3) +
     1                    DPENFA(IQ,4) )
130      CONTINUE
         UCOMPS = DPENFA(2,3)/DPENFA(1,3)
         VCOMPS = DPENFA(3,3)/DPENFA(1,3)

C        EAST  FACE

         PRESSE = 0.25*( PRESG2(KSE) + 2.*PRESSE + PRESG2(KNE) )
         DO 140 IQ = 1, NEQNFL
            DPENFA(IQ,5) = 0.25*( DPENFA(IQ,4) + 2.*DPENFA(IQ,5) +
     1                    DPENFA(IQ,6) )
140      CONTINUE
         UCOMPE = DPENFA(2,5)/DPENFA(1,5)
         VCOMPE = DPENFA(3,5)/DPENFA(1,5)

C        NORTH FACE

         PRESSN = 0.25*( PRESG2(KNW) + 2.*PRESSN + PRESG2(KNE) )
         DO 150 IQ = 1, NEQNFL
            DPENFA(IQ,7) = 0.25*( DPENFA(IQ,6) + 2.*DPENFA(IQ,7) +
     1                    DPENFA(IQ,8) )
150      CONTINUE
         UCOMPN = DPENFA(2,7)/DPENFA(1,7)
         VCOMPN = DPENFA(3,7)/DPENFA(1,7)

C        WEST  FACE
```

```
      PRESSW = 0.25*( PRESG2(KSW) + 2.*PRESSW + PRESG2(KNW) )
      DO 160 IQ = 1, NEQNFL
         DPENFA(IQ,9) = 0.25*( DPENFA(IQ,8) + 2.*DPENFA(IQ,9) +
     1                    DPENFA(IQ,2) )
160   CONTINUE
      UCOMPW = DPENFA(2,9)/DPENFA(1,9)
      VCOMPW = DPENFA(3,9)/DPENFA(1,9)

C     ----------
C     FLUX TERMS
C     ----------
C     SOUTH

      BIGFS(1) = DPENFA(2,3)
      BIGFS(2) = DPENFA(2,3)*UCOMPS + PRESSS
      BIGFS(3) = DPENFA(2,3)*VCOMPS
      BIGFS(4) = UCOMPS*(DPENFA(4,3) + PRESSS)

      BIGGS(1) = DPENFA(3,3)
      BIGGS(2) = BIGFS(3)
      BIGGS(3) = DPENFA(3,3)*VCOMPS + PRESSS
      BIGGS(4) = VCOMPS*(DPENFA(4,3) + PRESSS)

C     EAST

      BIGFE(1) = DPENFA(2,5)
      BIGFE(2) = DPENFA(2,5)*UCOMPE + PRESSE
      BIGFE(3) = DPENFA(2,5)*VCOMPE
      BIGFE(4) = UCOMPE*(DPENFA(4,5) + PRESSE)
      BIGGE(1) = DPENFA(3,5)
      BIGGE(2) = BIGFE(3)
      BIGGE(3) = DPENFA(3,5)*VCOMPE + PRESSE
      BIGGE(4) = VCOMPE*(DPENFA(4,5) + PRESSE)

C     NORTH

      BIGFN(1) = DPENFA(2,7)
      BIGFN(2) = DPENFA(2,7)*UCOMPN + PRESSN
      BIGFN(3) = DPENFA(2,7)*VCOMPN
      BIGFN(4) = UCOMPN*(DPENFA(4,7) + PRESSN)
      BIGGN(1) = DPENFA(3,7)
      BIGGN(2) = BIGFN(3)
      BIGGN(3) = DPENFA(3,7)*VCOMPN + PRESSN
      BIGGN(4) = VCOMPN*(DPENFA(4,7) + PRESSN)

C     WEST

      BIGFW(1) = DPENFA(2,9)
      BIGFW(2) = DPENFA(2,9)*UCOMPW + PRESSW
      BIGFW(3) = DPENFA(2,9)*VCOMPW
      BIGFW(4) = UCOMPW*(DPENFA(4,9) + PRESSW)
      BIGGW(1) = DPENFA(3,9)
      BIGGW(2) = BIGFW(3)
      BIGGW(3) = DPENFA(3,9)*VCOMPW + PRESSW
      BIGGW(4) = VCOMPW*(DPENFA(4,9) + PRESSW)
```

```
C          OTHER FLUX TERMS ASSOCIATED WITH CHEMISTRY

           DO 170 JS = NEQBAS+1, NEQNFL
              BIGFS(JS) = DPENFA(JS,3)*UCOMPS
              BIGGS(JS) = DPENFA(JS,3)*VCOMPS
              BIGFE(JS) = DPENFA(JS,5)*UCOMPE
              BIGGE(JS) = DPENFA(JS,5)*VCOMPE
              BIGFN(JS) = DPENFA(JS,7)*UCOMPN
              BIGGN(JS) = DPENFA(JS,7)*VCOMPN
              BIGFW(JS) = DPENFA(JS,9)*UCOMPW
              BIGGW(JS) = DPENFA(JS,9)*VCOMPW
170        CONTINUE

C          --------------
C          JACOBIAN TERMS
C          --------------
C
C          DEFINE DEPENDENT VARIABLES AT THE CENTER OF THE CELL

           DO 180 IQ = 1, NEQNFL
              DPENJA(IQ) = 0.25*( DPENFA(IQ,3) + DPENFA(IQ,5) +
     1                            DPENFA(IQ,7) + DPENFA(IQ,9) )
180        CONTINUE

C          SET UP THE QUANTITIES NEEDED TO COMPUTE SOURCE TERMS AND
C          JACOBIANS

           DO 190 IQ = NEQBAS+1, NEQNFL
              DELTA     = 0.001*DPENJA(IQ)
              IF (DELTA .EQ. 0.) DELTA = 0.001
              UTOP(IQ)  = DPENJA(IQ) + DELTA
              TOTAL(IQ) = DELTA
190        CONTINUE
C
C          NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
C          UCOMPR, VCOMPR, GAMAPR, YSPEPR ETC. AND GET THE SOURCE TERMS
C          FOR THE CELL
C
           CALL FRSOUR
C
           DO 200 JS = NEQBAS+1, NEQNFL
              BWCELL(JS) = BIGWJA(JS)
200        CONTINUE

           UCOMPC = UCOMPR
           VCOMPC = VCOMPR
           U2     = UCOMPR*UCOMPR
           V2     = VCOMPR*VCOMPR

           GM1    = GAMAPR - 1.
           GM3    = GM1 - 2.
           PAEBR  = (BEPSPR+PRESPR)/RHORPR

           FUJACO(1,2) = 1.
           GUJACO(1,3) = 1.
C
           GUJACO(2,1) = -UCOMPC*VCOMPC
```

```
      GUJACO(2,2) =  VCOMPC
      GUJACO(2,3) =  UCOMPC
C
      FUJACO(3,1) = GUJACO(2,1)
      FUJACO(3,2) = GUJACO(2,2)
      FUJACO(3,3) = GUJACO(2,3)
C
      FUJACO(2,1) = 0.5*(GM3*U2 + GM1*V2)
      FUJACO(2,2) = -GM3*UCOMPR
      FUJACO(2,3) = -GM1*VCOMPR
      FUJACO(2,4) =  GM1
C
      GUJACO(3,1) = FUJACO(2,1) - V2 + U2
      GUJACO(3,2) = FUJACO(2,2) - 2.*UCOMPC
      GUJACO(3,3) = FUJACO(2,3) - 2.*VCOMPC
      GUJACO(3,4) = FUJACO(2,4)
C
      FUJACO(4,1) = UCOMPR*(FUJACO(2,1) + U2 - PAEBR)
      FUJACO(4,2) = PAEBR - 2.*U2 + UCOMPR*FUJACO(2,2)
      FUJACO(4,3) = UCOMPR*FUJACO(2,3)
      FUJACO(4,4) = UCOMPR*(FUJACO(2,4) + 1.)
C
      GUJACO(4,1) = VCOMPR*(FUJACO(2,1) + U2 - PAEBR)
      GUJACO(4,2) = VCOMPR*(FUJACO(2,2) - 2.*UCOMPR)
      GUJACO(4,3) = VCOMPR*FUJACO(2,3) + PAEBR
      GUJACO(4,4) = VCOMPR*(FUJACO(2,4) + 1.)
C
      DO 210 JS = NEQBAS + 1, NEQNFL

         YS            = DPENJA(JS)/DPENJA(1)

         FUJACO(JS,1 ) = -UCOMPC*YS
         FUJACO(JS,2 ) =  YS
         FUJACO(JS,JS) =  UCOMPC

         GUJACO(JS,1 ) = -VCOMPC*YS
         GUJACO(JS,3 ) =  YS
         GUJACO(JS,JS) =  VCOMPC

210   CONTINUE

      F2BOT = DPENJA(2)*UCOMPC + PRESPR

      DO 220 JS   = NEQBAS + 1, NEQNFL
         WBOT(JS) = BIGWJA(JS)
220   CONTINUE

C     COMPUTE THE NUMERICAL JACOBIANS BY TAKING FORWARD DIFFERNCES

      DO 250 LS = NEQBAS+1, NEQNFL

C        COMPUTE VALUES AT TOP
         UDUMMY     = DPENJA(LS)
         DPENJA(LS) = UTOP(LS)
         CALL         E2SOUR
         F2TOP      = BGF2JA
```

```
            DO 230 JS   = NEQBAS + 1, NEQNFL
               WTOP(JS) = BIGWJA(JS)
230         - CONTINUE

C           RESET THE VALUE OF THE DEPENDENT VARIABLE

            DPENJA(LS) = UDUMMY

C           NOW TAKE FORWARD DIFFERENCES

            FUJACO(2,LS) = (F2TOP - F2BOT)/TOTAL(LS)
            GUJACO(3,LS) = FUJACO(2,LS)
            FUJACO(4,LS) = FUJACO(2,LS)*UCOMPC
            GUJACO(4,LS) = FUJACO(2,LS)*VCOMPC

            DO 240 JS = NEQBAS + 1, NEQNFL
               WUJACO(JS,LS) = (WTOP(JS) - WBOT(JS))/TOTAL(LS)
240         CONTINUE

250       CONTINUE

C         ------------------------------
C         FIRST ORDER CELL CHANGE DUCELL
C         ------------------------------

C         CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE

          DO 260 J = 1, NEQNFL
             DUCELL(J) = BWCELL(J)*CELLTI(ICELL) + DTDVOL*(
     1            BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1            BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1            BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
     1            BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )
260       CONTINUE

C         --------------------
C         JACOBIAN CHANGE BLOCK
C         --------------------

C         COMPUTE CHANGES DUE TO JACOBIANS

          DO 280 J = 1, NEQNFL
             DFCELL = 0.
             DGCELL = 0.
             DWCELL = 0.

             DO 270 K = 1, NEQNFL
                DFCELL = DFCELL + FUJACO(J,K)*DUCELL(K)
                DGCELL = DGCELL + GUJACO(J,K)*DUCELL(K)
                DWCELL = DWCELL + WUJACO(J,K)*DUCELL(K)
270          CONTINUE

C            TRANSFORM THE JACOBIAN CHANGES (ONLY FU AND GU) AND
C            MULTIPLY WITH THEIR RESPECTIVE SCALINGS OF TIME

             TEMPF  = DFCELL
             DFCELL = DTDVOL*( TEMPF*DYNSM2(ICELL)
```

```
     1                       - DGCELL*DXNSM2(ICELL))
               DGCELL = DTDVOL*(-TEMPF*DYEWM2(ICELL)
     1         -             + DGCELL*DXEWM2(ICELL))
               DWCELL = 0.5*CELLTI(ICELL)*DWCELL*IMPLTI

C              ---------------
C              DIFFUSION  TERMS
C              ---------------

C              COMPUTE THE DIFFUSION TERMS FOR THE FOUR EDGES

               SIGGSW = SIGGE2(KSW)*DPENG2(J,KSW)
               SIGGSE = SIGGE2(KSE)*DPENG2(J,KSE)
               SIGGNE = SIGGE2(KNE)*DPENG2(J,KNE)
               SIGGNW = SIGGE2(KNW)*DPENG2(J,KNW)

C              COMPUTE THE DIFFUSION TERM FOR THE WHOLE CELL

               SIGCEL= 0.25*(SIGGSW + SIGGSE + SIGGNE + SIGGNW)
               SIGGSW = SIGCEL - SIGGSW
               SIGGSE = SIGCEL - SIGGSE
               SIGGNE = SIGCEL - SIGGNE
               SIGGNW = SIGCEL - SIGGNW
C
               SIGGSW = DSDIFF*SIGGSW
               SIGGSE = DSDIFF*SIGGSE
               SIGGNE = DSDIFF*SIGGNE
               SIGGNW = DSDIFF*SIGGNW
CTEST
C              SIGGMX = 0.1*ABS(DUCELL(J))
C              SIGGMN = -SIGGMX
C              SIGGSW = MIN(SIGGSW,SIGGMX)
C              SIGGSE = MIN(SIGGSE,SIGGMX)
C              SIGGNE = MIN(SIGGNE,SIGGMX)
C              SIGGNW = MIN(SIGGNW,SIGGMX)
C              SIGGSW = MAX(SIGGSW,SIGGMN)
C              SIGGSE = MAX(SIGGSE,SIGGMN)
C              SIGGNE = MAX(SIGGNE,SIGGMN)
C              SIGGNW = MAX(SIGGNW,SIGGMN)
CTEST

C              ----------------------
C              COMPUTATION OF CHANGES
C              ----------------------

C              FOCIT IS DUCELL; FIND SOCIT AND CORNER CHANGES

               SOCITSW  = - DFCELL - DGCELL + DWCELL
               SOCITNW  = - DFCELL + DGCELL + DWCELL
               SOCITNE  = + DFCELL + DGCELL + DWCELL
               SOCITSE  = + DFCELL - DGCELL + DWCELL

               DELSW(J) = 0.25*( DUCELL(J) + SOCITSW + SIGGSW )
               DELNW(J) = 0.25*( DUCELL(J) + SOCITNW + SIGGNW )
               DELNE(J) = 0.25*( DUCELL(J) + SOCITNE + SIGGNE )
               DELSE(J) = 0.25*( DUCELL(J) + SOCITSE + SIGGSE )
```

```
280       CONTINUE

C         WANT TO DO IMPLICIT SOURCE TERMS ?

          IF (IMPLTI .EQ. 0) THEN
             CALL PTIMP2 (KSW, ICELL, DELSW)
             CALL PTIMP2 (KSE, ICELL, DELSE)
             CALL PTIMP2 (KNW, ICELL, DELNW)
             CALL PTIMP2 (KNE, ICELL, DELNE)
          ENDIF

C         ----------------------
C         DISTRIBUTION OF CHANGES
C         ----------------------

C         DISTRIBUTE CONVECTIVE AND DIFFUSIVE CHANGES

          DO 290 J = 1, NEQNFL

             DELS  = 0.5*( DELSE(J) + DELSW(J) )
             DELN  = 0.5*( DELNE(J) + DELNW(J) )
             DELW  = 0.5*( DELSW(J) + DELNW(J) )
             DELE  = 0.5*( DELSE(J) + DELNE(J) )

             CHNGE2(J,KSW) = CHNGE2(J,KSW) + DELSW(J)
             CHNGE2(J,KNW) = CHNGE2(J,KNW) + DELNW(J)
             CHNGE2(J,KSE) = CHNGE2(J,KSE) + DELSE(J)
             CHNGE2(J,KNE) = CHNGE2(J,KNE) + DELNE(J)

             IF(KN .NE. 0) CHNGE2(J,KN) = CHNGE2(J,KN) + DELN
             IF(KS .NE. 0) CHNGE2(J,KS) = CHNGE2(J,KS) + DELS
             IF(KW .NE. 0) CHNGE2(J,KW) = CHNGE2(J,KW) + DELW
             IF(KE .NE. 0) CHNGE2(J,KE) = CHNGE2(J,KE) + DELE

290       CONTINUE

300    CONTINUE
C
C                    ------------
C                    NOMENCLATURE
C                    ------------
C
C
C                         BIGFN      BIGGN
C
C                  KNW            KN            KNE
C                     +-----------+------------+
C                     |8          7           6|
C          B  B       |         ICELL          |       B  B
C          I  I       |                        |       I  I
C          G  G    KW +9          +1          5+ KE    G  G
C          G  F       |           KC           |       F  G
C          W  W       |                        |       E  E
C                     |2          3           4|
C                     +-----------+------------+
C                  KSW            KS            KSE
C
```

```
C                          BIGFS      BIGGS
C
      -
      RETURN
      END
```

E3SOLF

```
      SUBROUTINE E2SOLO (ITGL)
C                E3SOLF

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.inc     '
      INCLUDE '[.INC] JACOMN.INC/LIST'
      INCLUDE '[.INC] M2COMN.INC/LIST'
      INCLUDE '[.INC] PRCOMN.INC/LIST'
      INCLUDE '[.INC] TICOMN.INC/LIST'
      COMMON/WUCOMN/ WUJACO
      DIMENSION BIGFS (MEQNFL)         ,   BIGFE (MEQNFL)         ,
     1          BIGFN (MEQNFL)         ,   BIGFW (MEQNFL)         ,
     2          BIGGS (MEQNFL)         ,   BIGGE (MEQNFL)         ,
     3          BIGGN (MEQNFL)         ,   BIGGW (MEQNFL)         ,
     4          DELSW (MEQNFL)         ,   DELSE (MEQNFL)         ,
     5          DELNW (MEQNFL)         ,   DELNE (MEQNFL)         ,
     6          BWCELL(MEQNFL)         ,   DPENFA(MEQNFL,4)       ,
     7          FUJACO(MEQNFL,MEQNFL) ,   GUJACO(MEQNFL,MEQNFL) ,
     8          WUJACO(MEQNFL,MEQNFL) ,   DUCELL(MEQNFL)

      DIMENSION DVISC (MEQNFL)
      DIMENSION UTOP  (MEQNFL)         ,   TOTAL (MEQNFL)         ,
     1          WTOP  (MEQNFL)         ,   WBOT  (MEQNFL)

      DATA FUJACO /100*0./
      DATA GUJACO /100*0./
      DATA WUJACO /100*0./
      DATA BWCELL /10*0./

C*********************************************************************

C       THIS SUBROUTINE STEPS THROUGH EACH CELL ON THE SPATIAL LEVEL ITGL
C       AND APPLIES NI'S SCHEME, I.E., INTEGRATES OVER ALL CELLS ON ITGL.
C       IT ALSO COMPUTES THE ANALYTICAL AS WELL NUMERICAL JACOBIANS,
C       BECAUSE THEIR STORAGE IS COSTLY.  THIS SUBROUTINE CAN BE USED
C       FOR GRIDS WHICH HAVE NOT BEEN EMBEDDED YET.
C       DPENFA : VALUES OF DEPENDENT VARIABLES AT THE FACES
C       DPENG2 : VALUES OF DEPENDENT VARIABLES AT THE NODES
C       DPENJA : VALUES OF DEPENDENT VARIABLES FOR COMPUTING JACOBIANS
```

```
C       !!!!!!    THIS SUBROUTINE IS SPECIALIZED FOR MEQNFL=10     !!!!!!

C*************************************************************************
C
C       IMPLTI = 0 MEANS DO IMPLICIT SOURCE TERMS
C                1 MEANS DO EXPLICIT SOURCE TERMS
C                2 MEANS DO EXPLICIT SOURCE TERMS WITH FROZEN CHEMISTRY

        GOTO (310,10,610) IMPLTI+1
C       RETURN

C       USE EXPLICIT SOURCE TERMS
C       STEP THROUGH EACH CELL AT THIS LEVEL
C
CVD$    NOLSTVAL
10      DO 160 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C          ----------------------
C          CELL/NODE DETERMINATION
C          ----------------------

C          FIND THE CELL TO BE INTEGRATED

           ICELL = ICELTI(JCELL)

C          SET UP NODE POINTERS FOR THIS CELL

           KSW = ICELG2( 2,ICELL)
           KSE = ICELG2( 4,ICELL)
           KNE = ICELG2( 6,ICELL)
           KNW = ICELG2( 8,ICELL)

C          --------
C          GEOMETRY
C          --------
C
C          GEOMETRY OF ALL CELL CORNERS

           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)

           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)

           XNE = GEOMG2(1,KNE)
           YNE = GEOMG2(2,KNE)

           XNW = GEOMG2(1,KNW)
           YNW = GEOMG2(2,KNW)
C
C          THE RATIO DELTA-t TO CELL VOLUME
           DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

C          COMPUTE THE AVERAGE COEFFICIENT FOR DIFFUSION

           DSDIFF = 0.5*PERIM2(ICELL)*DTDVOL
```

```
C          -------------
C          FACIAL VALUES
C          -------------
C
C          COMPUTE THE DEPENDENT VARIABLES AT THE FACES

           PRESSS = 0.5*( PRESG2(KSW) + PRESG2(KSE) )
           PRESSE = 0.5*( PRESG2(KSE) + PRESG2(KNE) )
           PRESSN = 0.5*( PRESG2(KNW) + PRESG2(KNE) )
           PRESSW = 0.5*( PRESG2(KSW) + PRESG2(KNW) )

CVD$       NOLSTVAL
           DO 20 IQ = 1, NEQNFL
              DPENFA(IQ,1) = 0.5*( DPENG2(IQ,KSW) + DPENG2(IQ,KSE) )
              DPENFA(IQ,2) = 0.5*( DPENG2(IQ,KSE) + DPENG2(IQ,KNE) )
              DPENFA(IQ,3) = 0.5*( DPENG2(IQ,KNE) + DPENG2(IQ,KNW) )
              DPENFA(IQ,4) = 0.5*( DPENG2(IQ,KNW) + DPENG2(IQ,KSW) )
20         CONTINUE

           UCOMPS = DPENFA(2,1)/DPENFA(1,1)
           VCOMPS = DPENFA(3,1)/DPENFA(1,1)
           UCOMPE = DPENFA(2,2)/DPENFA(1,2)
           VCOMPE = DPENFA(3,2)/DPENFA(1,2)
           UCOMPN = DPENFA(2,3)/DPENFA(1,3)
           VCOMPN = DPENFA(3,3)/DPENFA(1,3)
           UCOMPW = DPENFA(2,4)/DPENFA(1,4)
           VCOMPW = DPENFA(3,4)/DPENFA(1,4)

C          ----------
C          FLUX TERMS
C          ----------
C          SOUTH

           BIGFS(1) = DPENFA(2,1)
           BIGFS(2) = DPENFA(2,1)*UCOMPS + PRESSS
           BIGFS(3) = DPENFA(2,1)*VCOMPS
           BIGFS(4) = UCOMPS*(DPENFA(4,1) + PRESSS)

           BIGGS(1) = DPENFA(3,1)
           BIGGS(2) = BIGFS(3)
           BIGGS(3) = DPENFA(3,1)*VCOMPS + PRESSS
           BIGGS(4) = VCOMPS*(DPENFA(4,1) + PRESSS)

C          EAST

           BIGFE(1) = DPENFA(2,2)
           BIGFE(2) = DPENFA(2,2)*UCOMPE + PRESSE
           BIGFE(3) = DPENFA(2,2)*VCOMPE
           BIGFE(4) = UCOMPE*(DPENFA(4,2) + PRESSE)
           BIGGE(1) = DPENFA(3,2)
           BIGGE(2) = BIGFE(3)
           BIGGE(3) = DPENFA(3,2)*VCOMPE + PRESSE
           BIGGE(4) = VCOMPE*(DPENFA(4,2) + PRESSE)

C          NORTH

           BIGFN(1) = DPENFA(2,3)
```

```
            BIGFN(2) = DPENFA(2,3)*UCOMPN + PRESSN
            BIGFN(3) = DPENFA(2,3)*VCOMPN
            BIGFN(4) = UCOMPN*(DPENFA(4,3) + PRESSN)
            BIGGN(1) = DPENFA(3,3)
            BIGGN(2) = BIGFN(3)
            BIGGN(3) = DPENFA(3,3)*VCOMPN + PRESSN
            BIGGN(4) = VCOMPN*(DPENFA(4,3) + PRESSN)

C           WEST

            BIGFW(1) = DPENFA(2,4)
            BIGFW(2) = DPENFA(2,4)*UCOMPW + PRESSW
            BIGFW(3) = DPENFA(2,4)*VCOMPW
            BIGFW(4) = UCOMPW*(DPENFA(4,4) + PRESSW)
            BIGGW(1) = DPENFA(3,4)
            BIGGW(2) = BIGFW(3)
            BIGGW(3) = DPENFA(3,4)*VCOMPW + PRESSW
            BIGGW(4) = VCOMPW*(DPENFA(4,4) + PRESSW)

C           OTHER FLUX TERMS ASSOCIATED WITH CHEMISTRY

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 30 JS = NEQBAS+1, NEQNFL
               BIGFS(JS) = DPENFA(JS,1)*UCOMPS
               BIGGS(JS) = DPENFA(JS,1)*VCOMPS
               BIGFE(JS) = DPENFA(JS,2)*UCOMPE
               BIGGE(JS) = DPENFA(JS,2)*VCOMPE
               BIGFN(JS) = DPENFA(JS,3)*UCOMPN
               BIGGN(JS) = DPENFA(JS,3)*VCOMPN
               BIGFW(JS) = DPENFA(JS,4)*UCOMPW
               BIGGW(JS) = DPENFA(JS,4)*VCOMPW
30          CONTINUE

C           --------------
C           JACOBIAN TERMS
C           --------------
C
C           DEFINE DEPENDENT VARIABLES AT THE CENTER OF THE CELL

CVD$        NOLSTVAL
            DO 40 IQ = 1, NEQNFL
               DPENJA(IQ) = 0.25*( DPENFA(IQ,1) + DPENFA(IQ,2) +
     1                             DPENFA(IQ,3) + DPENFA(IQ,4) )
40          CONTINUE

C           SET UP THE QUANTITIES NEEDED TO COMPUTE SOURCE TERMS AND
C           JACOBIANS

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 50 IQ = NEQBAS+1, NEQNFL
               DELTA     = 0.001*DPENJA(IQ)
               IF (DELTA .EQ. 0.) DELTA = 0.001
               UTOP(IQ)  = DPENJA(IQ) + DELTA
               TOTAL(IQ) = DELTA
50          CONTINUE
```

```
C
C          NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
C          UCOMPR, VCOMPR, GAMAPR, YSPEPR ETC. AND GET THE SOURCE TERMS
C          FOR THE CELL
C
           SONDPR = CELLTI(ICELL)
           CALL FRSOUR
C
CVD$       NOLSTVAL
CVD$       NOVECTOR
           DO 60 JS = NEQBAS+1, NEQNFL
              BWCELL(JS) = BIGWJA(JS)
60         CONTINUE

           UCOMPC = UCOMPR
           VCOMPC = VCOMPR
           U2     = UCOMPR*UCOMPR
           V2     = VCOMPR*VCOMPR

           GM1    = GAMAPR - 1.
           GM3    = GM1 - 2.
           PAEBR  = (BEPSPR+PRESPR)/RHORPR

           FUJACO(1,2) = 1.
           GUJACO(1,3) = 1.
C
           GUJACO(2,1) = -UCOMPC*VCOMPC
           GUJACO(2,2) =  VCOMPC
           GUJACO(2,3) =  UCOMPC
C
           FUJACO(3,1) = GUJACO(2,1)
           FUJACO(3,2) = GUJACO(2,2)
           FUJACO(3,3) = GUJACO(2,3)
C
           FUJACO(2,1) = 0.5*(GM3*U2 + GM1*V2)
           FUJACO(2,2) = -GM3*UCOMPR
           FUJACO(2,3) = -GM1*VCOMPR
           FUJACO(2,4) =  GM1
C
           GUJACO(3,1) = FUJACO(2,1) - V2 + U2
           GUJACO(3,2) = FUJACO(2,2) - 2.*UCOMPC
           GUJACO(3,3) = FUJACO(2,3) - 2.*VCOMPC
           GUJACO(3,4) = FUJACO(2,4)
C
           FUJACO(4,1) = UCOMPR*(FUJACO(2,1) + U2 - PAEBR)
           FUJACO(4,2) = PAEBR - 2.*U2 + UCOMPR*FUJACO(2,2)
           FUJACO(4,3) = UCOMPR*FUJACO(2,3)
           FUJACO(4,4) = UCOMPR*(FUJACO(2,4) + 1.)
C
           GUJACO(4,1) = VCOMPR*(FUJACO(2,1) + U2 - PAEBR)
           GUJACO(4,2) = VCOMPR*(FUJACO(2,2) - 2.*UCOMPR)
           GUJACO(4,3) = VCOMPR*FUJACO(2,3) + PAEBR
           GUJACO(4,4) = VCOMPR*(FUJACO(2,4) + 1.)
C
CVD$       NOLSTVAL
CVD$       NOVECTOR
CVD$       NODEPCHK
```

```
      DO 70 JS = NEQBAS + 1, NEQNFL

       - YSP           = DPENJA(JS)/DPENJA(1)

         FUJACO(JS,1 ) = -UCOMPC*YSP
         FUJACO(JS,2 ) =  YSP
         FUJACO(JS,JS) =  UCOMPC

         GUJACO(JS,1 ) = -VCOMPC*YSP
         GUJACO(JS,3 ) =  YSP
         GUJACO(JS,JS) =  VCOMPC

70    CONTINUE

      F2BOT = DPENJA(2)*UCOMPC + PRESPR

CVD$  NOLSTVAL
CVD$  NOVECTOR
      DO 80 JS   = NEQBAS + 1, NEQNFL
         WBOT(JS) = BIGWJA(JS)
80    CONTINUE

C     COMPUTE THE NUMERICAL JACOBIANS BY TAKING FORWARD DIFFERNCES

CVD$  NOLSTVAL
CVD$  NOVECTOR
      DO 110 LS = NEQBAS+1, NEQNFL

C        COMPUTE VALUES AT TOP
         UDUMMY     = DPENJA(LS)
         DPENJA(LS) = UTOP(LS)
         CALL         E2SOUR
         F2TOP      = BGF2JA

CVD$     NOLSTVAL
CVD$     NOVECTOR
         DO 90 JS   = NEQBAS + 1, NEQNFL
            WTOP(JS) = BIGWJA(JS)
90       CONTINUE

C        RESET THE VALUE OF THE DEPENDENT VARIABLE

         DPENJA(LS) = UDUMMY

C        NOW TAKE FORWARD DIFFERENCES

         FUJACO(2,LS) = (F2TOP - F2BOT)/TOTAL(LS)
         GUJACO(3,LS) = FUJACO(2,LS)
         FUJACO(4,LS) = FUJACO(2,LS)*UCOMPC
         GUJACO(4,LS) = FUJACO(2,LS)*VCOMPC

CVD$     NOLSTVAL
CVD$     NOVECTOR
         DO 100 JS = NEQBAS + 1, NEQNFL
            WUJACO(JS,LS) = (WTOP(JS) - WBOT(JS))/TOTAL(LS)
100      CONTINUE
```

```
110       CONTINUE

C         ----------------------------
C         FIRST ORDER CELL CHANGE DUCELL
C         ----------------------------

C         CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE

CVD$      NOLSTVAL
          DO 120 J = 1, NEQNFL
             DUCELL(J) = BWCELL(J)*CELLTI(ICELL) + DTDVOL*(
     1           BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1           BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1           BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
     1           BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )
120       CONTINUE

C         COMPUTE THE DISTANCES FOR THE CELL UNDER CONSIDERATION
          XSO     = 0.5*(XSW+XSE)
          XEO     = 0.5*(XSE+XNE)
          XNO     = 0.5*(XNE+XNW)
          XWO     = 0.5*(XNW+XSW)

          YSO     = 0.5*(YSW+YSE)
          YEO     = 0.5*(YSE+YNE)
          YNO     = 0.5*(YNE+YNW)
          YWO     = 0.5*(YNW+YSW)

C         COMPUTE THE VELOCITY COMPONENTS AT SPECIAL POINTERS FOR
C         VISCOUS CALCULATIONS
          USW     = DPENG2(2,KSW)/DPENG2(1,KSW)
          USE     = DPENG2(2,KSE)/DPENG2(1,KSE)
          UNE     = DPENG2(2,KNE)/DPENG2(1,KNE)
          UNW     = DPENG2(2,KNW)/DPENG2(1,KNW)

          USO     = 0.5*(USW+USE)
          UEO     = 0.5*(USE+UNE)
          UNO     = 0.5*(UNE+UNW)
          UWO     = 0.5*(UNW+USW)
          UCO     = 0.25*(USW+USE+UNE+UNW)

          US1     = 0.5*(USW+USO)
          US2     = 0.5*(USO+USE)
          UE1     = 0.5*(USE+UEO)
          UE2     = 0.5*(UEO+UNE)
          UN1     = 0.5*(UNE+UNO)
          UN2     = 0.5*(UNO+UNW)
          UW1     = 0.5*(UNW+UWO)
          UW2     = 0.5*(UWO+USW)

          VSW     = DPENG2(3,KSW)/DPENG2(1,KSW)
          VSE     = DPENG2(3,KSE)/DPENG2(1,KSE)
          VNE     = DPENG2(3,KNE)/DPENG2(1,KNE)
          VNW     = DPENG2(3,KNW)/DPENG2(1,KNW)

          VSO     = 0.5*(VSW+VSE)
          VEO     = 0.5*(VSE+VNE)
```

```
      VNO     = 0.5*(VNE+VNW)
      VWO     = 0.5*(VNW+VSW)
      VCO     = 0.25*(VSW+VSE+VNE+VNW)

      VS1     = 0.5*(VSW+VSO)
      VS2     = 0.5*(VSO+VSE)
      VE1     = 0.5*(VSE+VEO)
      VE2     = 0.5*(VEO+VNE)
      VN1     = 0.5*(VNE+VNO)
      VN2     = 0.5*(VNO+VNW)
      VW1     = 0.5*(VNW+VWO)
      VW2     = 0.5*(VWO+VSW)

C     COMPUTE THE TEMPERATURE FOR VISCOUS CALCULATIONS
      TSW     = TEMPG2(KSW)
      TSE     = TEMPG2(KSE)
      TNE     = TEMPG2(KNE)
      TNW     = TEMPG2(KNW)

      TSO     = 0.5*(TSW+TSE)
      TEO     = 0.5*(TSE+TNE)
      TNO     = 0.5*(TNE+TNW)
      TWO     = 0.5*(TNW+TSW)
      TCO     = 0.25*(TSW+TSE+TNE+TNW)

      TS1     = 0.5*(TSW+TSO)
      TS2     = 0.5*(TSO+TSE)
      TE1     = 0.5*(TSE+TEO)
      TE2     = 0.5*(TEO+TNE)
      TN1     = 0.5*(TNE+TNO)
      TN2     = 0.5*(TNO+TNW)
      TW1     = 0.5*(TNW+TWO)
      TW2     = 0.5*(TWO+TSW)

C     COMPUTE THE VELOCITY GRADIENTS FOR VISCOUS CALCULATIONS

      DUDXW   = 2.*RVOLM2(ICELL)*( UWO*(YSW-YNW) + UN2*(YNW-YNO)
     1                           + UCO*(YNO-YSO) + US1*(YSO-YSW) )
      DUDXN   = 2.*RVOLM2(ICELL)*( UW1*(YWO-YNW) + UNO*(YNW-YNE)
     1                           + UE2*(YNE-YEO) + UCO*(YEO-YWO) )
      DUDXE   = 2.*RVOLM2(ICELL)*( UCO*(YSO-YNO) + UN1*(YNO-YNE)
     1                           + UEO*(YNE-YSE) + US2*(YSE-YSO) )
      DUDXS   = 2.*RVOLM2(ICELL)*( UW2*(YSW-YWO) + UCO*(YWO-YEO)
     1                           + UE1*(YEO-YSE) + USO*(YSE-YSW) )

      DUDYW   =-2.*RVOLM2(ICELL)*( UWO*(XSW-XNW) + UN2*(XNW-XNO)
     1                           + UCO*(XNO-XSO) + US1*(XSO-XSW) )
      DUDYN   =-2.*RVOLM2(ICELL)*( UW1*(XWO-XNW) + UNO*(XNW-XNE)
     1                           + UE2*(XNE-XEO) + UCO*(XEO-XWO) )
      DUDYE   =-2.*RVOLM2(ICELL)*( UCO*(XSO-XNO) + UN1*(XNO-XNE)
     1                           + UEO*(XNE-XSE) + US2*(XSE-XSO) )
      DUDYS   =-2.*RVOLM2(ICELL)*( UW2*(XSW-XWO) + UCO*(XWO-XEO)
     1                           + UE1*(XEO-XSE) + USO*(XSE-XSW) )

      DVDXW   = 2.*RVOLM2(ICELL)*( VWO*(YSW-YNW) + VN2*(YNW-YNO)
     1                           + VCO*(YNO-YSO) + VS1*(YSO-YSW) )
      DVDXN   = 2.*RVOLM2(ICELL)*( VW1*(YWO-YNW) + VNO*(YNW-YNE)
```

```
     1                                + VE2*(YNE-YEO) + VCO*(YEO-YWO) )
          DVDXE   = 2.*RVOLM2(ICELL)*( VCO*(YSO-YNO) + VN1*(YNO-YNE)
     1     -                          + VEO*(YNE-YSE) + VS2*(YSE-YSO) )
          DVDXS   = 2.*RVOLM2(ICELL)*( VW2*(YSW-YWO) + VCO*(YWO-YEO)
     1                               + VE1*(YEO-YSE) + VSO*(YSE-YSW) )

          DVDYW   =-2.*RVOLM2(ICELL)*( VWO*(XSW-XNW) + VN2*(XNW-XNO)
     1                               + VCO*(XNO-XSO) + VS1*(XSO-XSW) )
          DVDYN   =-2.*RVOLM2(ICELL)*( VW1*(XWO-XNW) + VNO*(XNW-XNE)
     1                               + VE2*(XNE-XEO) + VCO*(XEO-XWO) )
          DVDYE   =-2.*RVOLM2(ICELL)*( VCO*(XSO-XNO) + VN1*(XNO-XNE)
     1                               + VEO*(XNE-XSE) + VS2*(XSE-XSO) )
          DVDYS   =-2.*RVOLM2(ICELL)*( VW2*(XSW-XWO) + VCO*(XWO-XEO)
     1                               + VE1*(XEO-XSE) + VSO*(XSE-XSW) )

C         COMPUTE THE TEMPERATURE GRADIENTS FOR VISCOUS CALCULATIONS

          DTDXW   = 2.*RVOLM2(ICELL)*( TWO*(YSW-YNW) + TN2*(YNW-YNO)
     1                               + TCO*(YNO-YSO) + TS1*(YSO-YSW) )
          DTDXN   = 2.*RVOLM2(ICELL)*( TW1*(YWO-YNW) + TNO*(YNW-YNE)
     1                               + TE2*(YNE-YEO) + TCO*(YEO-YWO) )
          DTDXE   = 2.*RVOLM2(ICELL)*( TCO*(YSO-YNO) + TN1*(YNO-YNE)
     1                               + TEO*(YNE-YSE) + TS2*(YSE-YSO) )
          DTDXS   = 2.*RVOLM2(ICELL)*( TW2*(YSW-YWO) + TCO*(YWO-YEO)
     1                               + TE1*(YEO-YSE) + TSO*(YSE-YSW) )

          DTDYW   =-2.*RVOLM2(ICELL)*( TWO*(XSW-XNW) + TN2*(XNW-XNO)
     1                               + TCO*(XNO-XSO) + TS1*(XSO-XSW) )
          DTDYN   =-2.*RVOLM2(ICELL)*( TW1*(XWO-XNW) + TNO*(XNW-XNE)
     1                               + TE2*(XNE-XEO) + TCO*(XEO-XWO) )
          DTDYE   =-2.*RVOLM2(ICELL)*( TCO*(XSO-XNO) + TN1*(XNO-XNE)
     1                               + TEO*(XNE-XSE) + TS2*(XSE-XSO) )
          DTDYS   =-2.*RVOLM2(ICELL)*( TW2*(XSW-XWO) + TCO*(XWO-XEO)
     1                               + TE1*(XEO-XSE) + TSO*(XSE-XSW) )

C         COMPUTE THE VISOCITY COEFFICIENT AS GIVEN BY THE POWER LAW
C         FOR VISCOUS CALCULATIONS
          AMSW     = TEMPG2(KSW)**OMEGE2
          AMSE     = TEMPG2(KSE)**OMEGE2
          AMNE     = TEMPG2(KNE)**OMEGE2
          AMNW     = TEMPG2(KNW)**OMEGE2

          AMSO     = 0.5*(AMSW+AMSE)
          AMEO     = 0.5*(AMSE+AMNE)
          AMNO     = 0.5*(AMNE+AMNW)
          AMWO     = 0.5*(AMNW+AMSW)

C         COMPUTE THE THERMAL CONDUCTIVITY AS GIVEN BY THE POWER LAW
C         TIMES THE GAMMA FACTOR FOR VISCOUS CALCULATIONS
          CNSO     = AMSO*GFACE2
          CNEO     = AMEO*GFACE2
          CNNO     = AMNO*GFACE2
          CNWO     = AMWO*GFACE2

C         COMPUTE THE VISOUS TERMS FOR MOMENTUM EQUATIONS

          AVISXX   = AMWO*(2.*DUDXW-DVDYW)*(YNW-YSW) +
```

```
     1                AMNO*(2.*DUDXN-DVDYN)*(YNE-YNW) +
     1                AMEO*(2.*DUDXE-DVDYE)*(YSE-YNE) +
     1      -         AMSO*(2.*DUDXS-DVDYS)*(YSW-YSE)
            AVISXY   = AMWO*(DUDYW+DVDXW)*(XNW-XSW) +
     1                AMNO*(DUDYN+DVDXN)*(XNE-XNW) +
     1                AMEO*(DUDYE+DVDXE)*(XSE-XNE) +
     1                AMSO*(DUDYS+DVDXS)*(XSW-XSE)

            AVISYX   = AMWO*(DUDYW+DVDXW)*(YNW-YSW) +
     1                AMNO*(DUDYN+DVDXN)*(YNE-YNW) +
     1                AMEO*(DUDYE+DVDXE)*(YSE-YNE) +
     1                AMSO*(DUDYS+DVDXS)*(YSW-YSE)

            AVISYY   = AMWO*(2.*DVDYW-DUDXW)*(XNW-XSW) +
     1                AMNO*(2.*DVDYN-DUDXN)*(XNE-XNW) +
     1                AMEO*(2.*DVDYE-DUDXE)*(XSE-XNE) +
     1                AMSO*(2.*DVDYS-DUDXS)*(XSW-XSE)

C           COMPUTE THE VISOUS TERMS FOR ENERGY EQUATIONS

            AENEX1   = AMWO*UWO*(2.*DUDXW-DVDYW)*(YNW-YSW) +
     1                AMNO*UNO*(2.*DUDXN-DVDYN)*(YNE-YNW) +
     1                AMEO*UEO*(2.*DUDXE-DVDYE)*(YSE-YNE) +
     1                AMSO*USO*(2.*DUDXS-DVDYS)*(YSW-YSE)

            AENEX2   = AMWO*VWO*(DUDYW+DVDXW)*(YNW-YSW) +
     1                AMNO*VNO*(DUDYN+DVDXN)*(YNE-YNW) +
     1                AMEO*VEO*(DUDYE+DVDXE)*(YSE-YNE) +
     1                AMSO*VSO*(DUDYS+DVDXS)*(YSW-YSE)

            AENEX3   = RPRNE2*( CNWO*DTDXW*(YNW-YSW) +
     1                          CNNO*DTDXN*(YNE-YNW) +
     1                          CNEO*DTDXE*(YSE-YNE) +
     1                          CNSO*DTDXS*(YSW-YSE) )

            AENEY1   = AMWO*UWO*(DUDYW+DVDXW)*(XNW-XSW) +
     1                AMNO*UNO*(DUDYN+DVDXN)*(XNE-XNW) +
     1                AMEO*UEO*(DUDYE+DVDXE)*(XSE-XNE) +
     1                AMSO*USO*(DUDYS+DVDXS)*(XSW-XSE)

            AENEY2   = AMWO*VWO*(2.*DVDYW-DUDXW)*(XNW-XSW) +
     1                AMNO*VNO*(2.*DVDYN-DUDXN)*(XNE-XNW) +
     1                AMEO*VEO*(2.*DVDYE-DUDXE)*(XSE-XNE) +
     1                AMSO*VSO*(2.*DVDYS-DUDXS)*(XSW-XSE)

            AENEY3   = RPRNE2*( CNWO*DTDYW*(XNW-XSW) +
     1                          CNNO*DTDYN*(XNE-XNW) +
     1                          CNEO*DTDYE*(XSE-XNE) +
     1                          CNSO*DTDYS*(XSW-XSE) )

            TFACTOR = -RREYE2*DTDVOL

            DVISC(1) = 0.
            DVISC(2) = TFACTOR*(2./3.*AVISXX - AVISXY)
            DVISC(3) = TFACTOR*(AVISYX - 2./3.*AVISYY)
            DVISC(4) = TFACTOR*((2./3.*AENEX1+AENEX2+AENEX3) -
     1                          (2./3.*AENEY2+AENEY1+AENEY3) )
```

```
C          COMPUTE THE VISCOUS TERMS PERTAINING TO SPECIES EQUATIONS
           DO 125 J = NEQBAS + 1, NEQNFL
              AYSW  = DPENG2(J,KSW)/DPENG2(1,KSW)
              AYSE  = DPENG2(J,KSE)/DPENG2(1,KSE)
              AYNE  = DPENG2(J,KNE)/DPENG2(1,KNE)
              AYNW  = DPENG2(J,KNW)/DPENG2(1,KNW)
              AYSO  = 0.5*(AYSW+AYSE)
              AYEO  = 0.5*(AYSE+AYNE)
              AYNO  = 0.5*(AYNE+AYNW)
              AYWO  = 0.5*(AYNW+AYSW)
              AYS1  = 0.5*(AYSW+AYSO)
              AYS2  = 0.5*(AYSO+AYSE)
              AYE1  = 0.5*(AYSE+AYEO)
              AYE2  = 0.5*(AYEO+AYNE)
              AYN1  = 0.5*(AYNE+AYNO)
              AYN2  = 0.5*(AYNO+AYNW)
              AYW1  = 0.5*(AYNW+AYWO)
              AYW2  = 0.5*(AYWO+AYSW)

              DADXW = 2.*RVOLM2(ICELL)*(AYWO*(YSW-YNW)
     1               +AYN2*(YNW-YNO) +AYCO*(YNO-YSO) +AYS1*(YSO-YSW))
              DADXN = 2.*RVOLM2(ICELL)*(AYW1*(YWO-YNW)
     1               +AYNO*(YNW-YNE) +AYE2*(YNE-YEO) +AYCO*(YEO-YWO))
              DADXE = 2.*RVOLM2(ICELL)*(AYCO*(YSO-YNO)
     1               +AYN1*(YNO-YNE) +AYEO*(YNE-YSE) +AYS2*(YSE-YSO))
              DADXS = 2.*RVOLM2(ICELL)*(AYW2*(YSW-YWO)
     1               +AYCO*(YWO-YEO) +AYE1*(YEO-YSE) +AYSO*(YSE-YSW))

              DADYW =-2.*RVOLM2(ICELL)*(AYWO*(XSW-XNW)
     1               +AYN2*(XNW-XNO) +AYCO*(XNO-XSO) +AYS1*(XSO-XSW))
              DADYN =-2.*RVOLM2(ICELL)*(AYW1*(XWO-XNW)
     1               +AYNO*(XNW-XNE) +AYE2*(XNE-XEO) +AYCO*(XEO-XWO))
              DADYE =-2.*RVOLM2(ICELL)*(AYCO*(XSO-XNO)
     1               +AYN1*(XNO-XNE) +AYEO*(XNE-XSE) +AYS2*(XSE-XSO))
              DADYS =-2.*RVOLM2(ICELL)*(AYW2*(XSW-XWO)
     1               +AYCO*(XWO-XEO) +AYE1*(XEO-XSE) +AYSO*(XSE-XSW))


              ADIFX   = AMWO*DADXW*(YNW-YSW) + AMNO*DADXN*(YNE-YNW) +
     1                  AMEO*DADXE*(YSE-YNE) + AMSO*DADXS*(YSW-YSE)
              ADIFY   = AMWO*DADYW*(XNW-XSW) + AMNO*DADYN*(XNE-XNW) +
     1                  AMEO*DADYE*(XSE-XNE) + AMSO*DADYS*(XSW-XSE)

              DVISC(J) = TFACTOR*RSCHE2*(ADIFX - ADIFY)

125        CONTINUE

C          ---------------------
C          JACOBIAN CHANGE BLOCK
C          ---------------------

C          COMPUTE CHANGES DUE TO JACOBIANS

CVD$       NOLSTVAL
           DO 140 J = 1, NEQNFL
              DFCELL = 0.
```

```
                DGCELL = 0.
                DWCELL = 0.

                DO 130 K = 1, NEQNFL
                   DFCELL = DFCELL + FUJACO(J,K)*DUCELL(K)
                   DGCELL = DGCELL + GUJACO(J,K)*DUCELL(K)
                   DWCELL = DWCELL + WUJACO(J,K)*DUCELL(K)
130             CONTINUE

C               TRANSFORM THE JACOBIAN CHANGES (ONLY FU AND GU) AND
C               MULTIPLY WITH THEIR RESPECTIVE SCALINGS OF TIME

                TEMPF  = DFCELL
                DFCELL = DTDVOL*( TEMPF*DYNSM2(ICELL)
     1                          -DGCELL*DXNSM2(ICELL))
                DGCELL = DTDVOL*(-TEMPF*DYEWM2(ICELL)
     1                          +DGCELL*DXEWM2(ICELL))
                DWCELL = 0.5*CELLTI(ICELL)*DWCELL

C               ---------------
C               DIFFUSION  TERMS
C               ---------------

C               COMPUTE THE DIFFUSION TERMS FOR THE FOUR EDGES

                SIGGSW = SIGGE2(KSW)*DPENG2(J,KSW)
                SIGGSE = SIGGE2(KSE)*DPENG2(J,KSE)
                SIGGNE = SIGGE2(KNE)*DPENG2(J,KNE)
                SIGGNW = SIGGE2(KNW)*DPENG2(J,KNW)

C               COMPUTE THE DIFFUSION TERM FOR THE WHOLE CELL

                SIGCEL= 0.25*(SIGGSW + SIGGSE + SIGGNE + SIGGNW)
                SIGGSW = SIGCEL - SIGGSW
                SIGGSE = SIGCEL - SIGGSE
                SIGGNE = SIGCEL - SIGGNE
                SIGGNW = SIGCEL - SIGGNW
C
                SIGGSW = DSDIFF*SIGGSW
                SIGGSE = DSDIFF*SIGGSE
                SIGGNE = DSDIFF*SIGGNE
                SIGGNW = DSDIFF*SIGGNW

C               ---------------------
C               COMPUTATION OF CHANGES
C               ---------------------

C               FOCIT IS DUCELL; FIND SOCIT AND CORNER CHANGES

                SOCITSW  = - DFCELL - DGCELL + DWCELL
                SOCITNW  = - DFCELL + DGCELL + DWCELL
                SOCITNE  = + DFCELL + DGCELL + DWCELL
                SOCITSE  = + DFCELL - DGCELL + DWCELL

                DELSW(J) = 0.25*(DUCELL(J) +DVISC(J) +SOCITSW +SIGGSW)
                DELNW(J) = 0.25*(DUCELL(J) +DVISC(J) +SOCITNW +SIGGNW)
                DELNE(J) = 0.25*(DUCELL(J) +DVISC(J) +SOCITNE +SIGGNE)
```

```
            DELSE(J) = 0.25*(DUCELL(J) +DVISC(J) +SOCITSE +SIGGSE)

140         CONTINUE

C           ----------------------
C           DISTRIBUTION OF CHANGES
C           ----------------------

C           DISTRIBUTE CONVECTIVE AND DIFFUSIVE CHANGES

CVD$        NOLSTVAL
            DO 150 J = 1, NEQNFL
               CHNGE2(J,KSW) = CHNGE2(J,KSW) + DELSW(J)
               CHNGE2(J,KNW) = CHNGE2(J,KNW) + DELNW(J)
               CHNGE2(J,KSE) = CHNGE2(J,KSE) + DELSE(J)
               CHNGE2(J,KNE) = CHNGE2(J,KNE) + DELNE(J)
150         CONTINUE

160      CONTINUE
C
         RETURN

C        USE IMPLICIT SOURCE TERMS
C        STEP THROUGH EACH CELL AT THIS LEVEL
C
CVD$     NOLSTVAL
310      DO 560 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C           ----------------------
C           CELL/NODE DETERMINATION
C           ----------------------

C           FIND THE CELL TO BE INTEGRATED

            ICELL = ICELTI(JCELL)

C           SET UP NODE POINTERS FOR THIS CELL

            KSW = ICELG2( 2,ICELL)
            KSE = ICELG2( 4,ICELL)
            KNE = ICELG2( 6,ICELL)
            KNW = ICELG2( 8,ICELL)

C           --------
C           GEOMETRY
C           --------
C
C           GEOMETRY OF ALL CELL CORNERS

            XSW = GEOMG2(1,KSW)
            YSW = GEOMG2(2,KSW)

            XSE = GEOMG2(1,KSE)
            YSE = GEOMG2(2,KSE)

            XNE = GEOMG2(1,KNE)
            YNE = GEOMG2(2,KNE)
```

```
      XNW = GEOMG2(1,KNW)
      YNW = GEOMG2(2,KNW)
C
C     THE RATIO DELTA-t TO CELL VOLUME
      DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

C     COMPUTE THE AVERAGE COEFFICIENT FOR DIFFUSION

      DSDIFF = 0.5*PERIM2(ICELL)*DTDVOL

C     -------------
C     FACIAL VALUES
C     -------------
C
C     COMPUTE THE DEPENDENT VARIABLES AT THE FACES

      PRESSS = 0.5*( PRESG2(KSW) + PRESG2(KSE) )
      PRESSE = 0.5*( PRESG2(KSE) + PRESG2(KNE) )
      PRESSN = 0.5*( PRESG2(KNW) + PRESG2(KNE) )
      PRESSW = 0.5*( PRESG2(KSW) + PRESG2(KNW) )

CVD$  NOLSTVAL
      DO 320 IQ = 1, NEQNFL
         DPENFA(IQ,1) = 0.5*( DPENG2(IQ,KSW) + DPENG2(IQ,KSE) )
         DPENFA(IQ,2) = 0.5*( DPENG2(IQ,KSE) + DPENG2(IQ,KNE) )
         DPENFA(IQ,3) = 0.5*( DPENG2(IQ,KNE) + DPENG2(IQ,KNW) )
         DPENFA(IQ,4) = 0.5*( DPENG2(IQ,KNW) + DPENG2(IQ,KSW) )
320   CONTINUE

      UCOMPS = DPENFA(2,1)/DPENFA(1,1)
      VCOMPS = DPENFA(3,1)/DPENFA(1,1)
      UCOMPE = DPENFA(2,2)/DPENFA(1,2)
      VCOMPE = DPENFA(3,2)/DPENFA(1,2)
      UCOMPN = DPENFA(2,3)/DPENFA(1,3)
      VCOMPN = DPENFA(3,3)/DPENFA(1,3)
      UCOMPW = DPENFA(2,4)/DPENFA(1,4)
      VCOMPW = DPENFA(3,4)/DPENFA(1,4)

C     ----------
C     FLUX TERMS
C     ----------
C     SOUTH

      BIGFS(1) = DPENFA(2,1)
      BIGFS(2) = DPENFA(2,1)*UCOMPS + PRESSS
      BIGFS(3) = DPENFA(2,1)*VCOMPS
      BIGFS(4) = UCOMPS*(DPENFA(4,1) + PRESSS)

      BIGGS(1) = DPENFA(3,1)
      BIGGS(2) = BIGFS(3)
      BIGGS(3) = DPENFA(3,1)*VCOMPS + PRESSS
      BIGGS(4) = VCOMPS*(DPENFA(4,1) + PRESSS)

C     EAST

      BIGFE(1) = DPENFA(2,2)
```

```
            BIGFE(2) = DPENFA(2,2)*UCOMPE + PRESSE
            BIGFE(3) = DPENFA(2,2)*VCOMPE
            BIGFE(4) = UCOMPE*(DPENFA(4,2) + PRESSE)
            BIGGE(1) = DPENFA(3,2)
            BIGGE(2) = BIGFE(3)
            BIGGE(3) = DPENFA(3,2)*VCOMPE + PRESSE
            BIGGE(4) = VCOMPE*(DPENFA(4,2) + PRESSE)

C           NORTH

            BIGFN(1) = DPENFA(2,3)
            BIGFN(2) = DPENFA(2,3)*UCOMPN + PRESSN
            BIGFN(3) = DPENFA(2,3)*VCOMPN
            BIGFN(4) = UCOMPN*(DPENFA(4,3) + PRESSN)
            BIGGN(1) = DPENFA(3,3)
            BIGGN(2) = BIGFN(3)
            BIGGN(3) = DPENFA(3,3)*VCOMPN + PRESSN
            BIGGN(4) = VCOMPN*(DPENFA(4,3) + PRESSN)

C           WEST

            BIGFW(1) = DPENFA(2,4)
            BIGFW(2) = DPENFA(2,4)*UCOMPW + PRESSW
            BIGFW(3) = DPENFA(2,4)*VCOMPW
            BIGFW(4) = UCOMPW*(DPENFA(4,4) + PRESSW)
            BIGGW(1) = DPENFA(3,4)
            BIGGW(2) = BIGFW(3)
            BIGGW(3) = DPENFA(3,4)*VCOMPW + PRESSW
            BIGGW(4) = VCOMPW*(DPENFA(4,4) + PRESSW)

C           OTHER FLUX TERMS ASSOCIATED WITH CHEMISTRY

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 330 JS = NEQBAS+1, NEQNFL
               BIGFS(JS) = DPENFA(JS,1)*UCOMPS
               BIGGS(JS) = DPENFA(JS,1)*VCOMPS
               BIGFE(JS) = DPENFA(JS,2)*UCOMPE
               BIGGE(JS) = DPENFA(JS,2)*VCOMPE
               BIGFN(JS) = DPENFA(JS,3)*UCOMPN
               BIGGN(JS) = DPENFA(JS,3)*VCOMPN
               BIGFW(JS) = DPENFA(JS,4)*UCOMPW
               BIGGW(JS) = DPENFA(JS,4)*VCOMPW
330         CONTINUE

C           --------------
C           JACOBIAN TERMS
C           --------------
C
C           DEFINE DEPENDENT VARIABLES AT THE CENTER OF THE CELL

CVD$        NOLSTVAL
            DO 340 IQ = 1, NEQNFL
               DPENJA(IQ) = 0.25*( DPENFA(IQ,1) + DPENFA(IQ,2) +
     1                             DPENFA(IQ,3) + DPENFA(IQ,4) )
340         CONTINUE
```

```
C          SET UP THE QUANTITIES NEEDED TO COMPUTE SOURCE TERMS AND
C          JACOBIANS
CVD$       NOLSTVAL
CVD$       NOVECTOR
           DO 350 IQ = NEQBAS+1, NEQNFL
              DELTA     = 0.001*DPENJA(IQ)
              IF (DELTA .EQ. 0.) DELTA = 0.001
              UTOP(IQ)  = DPENJA(IQ) + DELTA
              TOTAL(IQ) = DELTA
350        CONTINUE
C
C          NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
C          UCOMPR, VCOMPR, GAMAPR, YSPEPR ETC. AND GET THE SOURCE TERMS
C          FOR THE CELL
C
           SONDPR = CELLTI(ICELL)
           CALL FRSOUR
C
CVD$       NOLSTVAL
CVD$       NOVECTOR
           DO 360 JS = NEQBAS+1, NEQNFL
              BWCELL(JS) = BIGWJA(JS)
360        CONTINUE

           UCOMPC = UCOMPR
           VCOMPC = VCOMPR
           U2     = UCOMPR*UCOMPR
           V2     = VCOMPR*VCOMPR

           GM1    = GAMAPR - 1.
           GM3    = GM1 - 2.
           PAEBR  = (BEPSPR+PRESPR)/RHORPR

           FUJACO(1,2) = 1.
           GUJACO(1,3) = 1.
C
           GUJACO(2,1) = -UCOMPC*VCOMPC
           GUJACO(2,2) =  VCOMPC
           GUJACO(2,3) =  UCOMPC
C
           FUJACO(3,1) = GUJACO(2,1)
           FUJACO(3,2) = GUJACO(2,2)
           FUJACO(3,3) = GUJACO(2,3)
C
           FUJACO(2,1) = 0.5*(GM3*U2 + GM1*V2)
           FUJACO(2,2) = -GM3*UCOMPR
           FUJACO(2,3) = -GM1*VCOMPR
           FUJACO(2,4) =  GM1
C
           GUJACO(3,1) = FUJACO(2,1) - V2 + U2
           GUJACO(3,2) = FUJACO(2,2) - 2.*UCOMPC
           GUJACO(3,3) = FUJACO(2,3) - 2.*VCOMPC
           GUJACO(3,4) = FUJACO(2,4)
C
           FUJACO(4,1) = UCOMPR*(FUJACO(2,1) + U2 - PAEBR)
           FUJACO(4,2) = PAEBR - 2.*U2 + UCOMPR*FUJACO(2,2)
```

```
            FUJACO(4,3) = UCOMPR*FUJACO(2,3)
            FUJACO(4,4) = UCOMPR*(FUJACO(2,4) + 1.)
C           -
            GUJACO(4,1) = VCOMPR*(FUJACO(2,1) + U2 - PAEBR)
            GUJACO(4,2) = VCOMPR*(FUJACO(2,2) - 2.*UCOMPR)
            GUJACO(4,3) = VCOMPR*FUJACO(2,3) + PAEBR
            GUJACO(4,4) = VCOMPR*(FUJACO(2,4) + 1.)
C
CVD$        NOLSTVAL
CVD$        NOVECTOR
CVD$        NODEPCHK
            DO 370 JS = NEQBAS + 1, NEQNFL

               YSP           = DPENJA(JS)/DPENJA(1)

               FUJACO(JS,1 ) = -UCOMPC*YSP
               FUJACO(JS,2 ) =  YSP
               FUJACO(JS,JS) =  UCOMPC

               GUJACO(JS,1 ) = -VCOMPC*YSP
               GUJACO(JS,3 ) =  YSP
               GUJACO(JS,JS) =  VCOMPC

370         CONTINUE

            F2BOT = DPENJA(2)*UCOMPC + PRESPR

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 380 JS   = NEQBAS + 1, NEQNFL
               WBOT(JS) = BIGWJA(JS)
380         CONTINUE

C           COMPUTE THE NUMERICAL JACOBIANS BY TAKING FORWARD DIFFERNCES

CVD$        NOLSTVAL
CVD$        NOVECTOR
            DO 410 LS = NEQBAS+1, NEQNFL

C              COMPUTE VALUES AT TOP
               UDUMMY     = DPENJA(LS)
               DPENJA(LS) = UTOP(LS)
               CALL         E2SOUR
               F2TOP      = BGF2JA

CVD$           NOLSTVAL
CVD$           NOVECTOR
               DO 390 JS   = NEQBAS + 1, NEQNFL
                  WTOP(JS) = BIGWJA(JS)
390            CONTINUE

C              RESET THE VALUE OF THE DEPENDENT VARIABLE

               DPENJA(LS) = UDUMMY

C              NOW TAKE FORWARD DIFFERENCES
```

```
                  FUJACO(2,LS) = (F2TOP - F2BOT)/TOTAL(LS)
                - GUJACO(3,LS) = FUJACO(2,LS)
                  FUJACO(4,LS) = FUJACO(2,LS)*UCOMPC
                  GUJACO(4,LS) = FUJACO(2,LS)*VCOMPC

CVD$              NOLSTVAL
CVD$              NOVECTOR
                  DO 400 JS = NEQBAS + 1, NEQNFL
                     WUJACO(JS,LS) = (WTOP(JS) - WBOT(JS))/TOTAL(LS)
400               CONTINUE

410           CONTINUE

C             ------------------------------
C             FIRST ORDER CELL CHANGE DUCELL
C             ------------------------------

C             CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE

CVD$          NOLSTVAL
              DO 420 J = 1, NEQNFL
                 DUCELL(J) = BWCELL(J)*CELLTI(ICELL) + DTDVOL*(
     1                       BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1                       BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1                       BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
     1                       BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )
420           CONTINUE
C             COMPUTE THE DISTANCES FOR THE CELL UNDER CONSIDERATION
              XSO     = 0.5*(XSW+XSE)
              XEO     = 0.5*(XSE+XNE)
              XNO     = 0.5*(XNE+XNW)
              XWO     = 0.5*(XNW+XSW)

              YSO     = 0.5*(YSW+YSE)
              YEO     = 0.5*(YSE+YNE)
              YNO     = 0.5*(YNE+YNW)
              YWO     = 0.5*(YNW+YSW)

C             COMPUTE THE VELOCITY COMPONENTS AT SPECIAL POINTERS FOR
C             VISCOUS CALCULATIONS
              USW     = DPENG2(2,KSW)/DPENG2(1,KSW)
              USE     = DPENG2(2,KSE)/DPENG2(1,KSE)
              UNE     = DPENG2(2,KNE)/DPENG2(1,KNE)
              UNW     = DPENG2(2,KNW)/DPENG2(1,KNW)

              USO     = 0.5*(USW+USE)
              UEO     = 0.5*(USE+UNE)
              UNO     = 0.5*(UNE+UNW)
              UWO     = 0.5*(UNW+USW)
              UCO     = 0.25*(USW+USE+UNE+UNW)

              US1     = 0.5*(USW+USO)
              US2     = 0.5*(USO+USE)
              UE1     = 0.5*(USE+UEO)
              UE2     = 0.5*(UEO+UNE)
              UN1     = 0.5*(UNE+UNO)
              UN2     = 0.5*(UNO+UNW)
```

```
      UW1     = 0.5*(UNW+UWO)
      UW2     = 0.5*(UWO+USW)

      VSW     = DPENG2(3,KSW)/DPENG2(1,KSW)
      VSE     = DPENG2(3,KSE)/DPENG2(1,KSE)
      VNE     = DPENG2(3,KNE)/DPENG2(1,KNE)
      VNW     = DPENG2(3,KNW)/DPENG2(1,KNW)

      VSO     = 0.5*(VSW+VSE)
      VEO     = 0.5*(VSE+VNE)
      VNO     = 0.5*(VNE+VNW)
      VWO     = 0.5*(VNW+VSW)
      VCO     = 0.25*(VSW+VSE+VNE+VNW)

      VS1     = 0.5*(VSW+VSO)
      VS2     = 0.5*(VSO+VSE)
      VE1     = 0.5*(VSE+VEO)
      VE2     = 0.5*(VEO+VNE)
      VN1     = 0.5*(VNE+VNO)
      VN2     = 0.5*(VNO+VNW)
      VW1     = 0.5*(VNW+VWO)
      VW2     = 0.5*(VWO+VSW)

C     COMPUTE THE TEMPERATURE FOR VISCOUS CALCULATIONS
      TSW     = TEMPG2(KSW)
      TSE     = TEMPG2(KSE)
      TNE     = TEMPG2(KNE)
      TNW     = TEMPG2(KNW)

      TSO     = 0.5*(TSW+TSE)
      TEO     = 0.5*(TSE+TNE)
      TNO     = 0.5*(TNE+TNW)
      TWO     = 0.5*(TNW+TSW)
      TCO     = 0.25*(TSW+TSE+TNE+TNW)

      TS1     = 0.5*(TSW+TSO)
      TS2     = 0.5*(TSO+TSE)
      TE1     = 0.5*(TSE+TEO)
      TE2     = 0.5*(TEO+TNE)
      TN1     = 0.5*(TNE+TNO)
      TN2     = 0.5*(TNO+TNW)
      TW1     = 0.5*(TNW+TWO)
      TW2     = 0.5*(TWO+TSW)

C     COMPUTE THE VELOCITY GRADIENTS FOR VISCOUS CALCULATIONS

      DUDXW   = 2.*RVOLM2(ICELL)*( UWO*(YSW-YNW) + UN2*(YNW-YNO)
     1                           + UCO*(YNO-YSO) + US1*(YSO-YSW) )
      DUDXN   = 2.*RVOLM2(ICELL)*( UW1*(YWO-YNW) + UNO*(YNW-YNE)
     1                           + UE2*(YNE-YEO) + UCO*(YEO-YWO) )
      DUDXE   = 2.*RVOLM2(ICELL)*( UCO*(YSO-YNO) + UN1*(YNO-YNE)
     1                           + UEO*(YNE-YSE) + US2*(YSE-YSO) )
      DUDXS   = 2.*RVOLM2(ICELL)*( UW2*(YSW-YWO) + UCO*(YWO-YEO)
     1                           + UE1*(YEO-YSE) + USO*(YSE-YSW) )

      DUDYW   =-2.*RVOLM2(ICELL)*( UWO*(XSW-XNW) + UN2*(XNW-XNO)
     1                           + UCO*(XNO-XSO) + US1*(XSO-XSW) )
```

```
      DUDYN   =-2.*RVOLM2(ICELL)*( UW1*(XWO-XNW) + UNO*(XNW-XNE)
     1                           + UE2*(XNE-XEO) + UCO*(XEO-XWO) )
      DUDYE   =-2.*RVOLM2(ICELL)*( UCO*(XSO-XNO) + UN1*(XNO-XNE)
     1                           + UEO*(XNE-XSE) + US2*(XSE-XSO) )
      DUDYS   =-2.*RVOLM2(ICELL)*( UW2*(XSW-XWO) + UCO*(XWO-XEO)
     1                           + UE1*(XEO-XSE) + USO*(XSE-XSW) )

      DVDXW   = 2.*RVOLM2(ICELL)*( VWO*(YSW-YNW) + VN2*(YNW-YNO)
     1                           + VCO*(YNO-YSO) + VS1*(YSO-YSW) )
      DVDXN   = 2.*RVOLM2(ICELL)*( VW1*(YWO-YNW) + VNO*(YNW-YNE)
     1                           + VE2*(YNE-YEO) + VCO*(YEO-YWO) )
      DVDXE   = 2.*RVOLM2(ICELL)*( VCO*(YSO-YNO) + VN1*(YNO-YNE)
     1                           + VEO*(YNE-YSE) + VS2*(YSE-YSO) )
      DVDXS   = 2.*RVOLM2(ICELL)*( VW2*(YSW-YWO) + VCO*(YWO-YEO)
     1                           + VE1*(YEO-YSE) + VSO*(YSE-YSW) )

      DVDYW   =-2.*RVOLM2(ICELL)*( VWO*(XSW-XNW) + VN2*(XNW-XNO)
     1                           + VCO*(XNO-XSO) + VS1*(XSO-XSW) )
      DVDYN   =-2.*RVOLM2(ICELL)*( VW1*(XWO-XNW) + VNO*(XNW-XNE)
     1                           + VE2*(XNE-XEO) + VCO*(XEO-XWO) )
      DVDYE   =-2.*RVOLM2(ICELL)*( VCO*(XSO-XNO) + VN1*(XNO-XNE)
     1                           + VEO*(XNE-XSE) + VS2*(XSE-XSO) )
      DVDYS   =-2.*RVOLM2(ICELL)*( VW2*(XSW-XWO) + VCO*(XWO-XEO)
     1                           + VE1*(XEO-XSE) + VSO*(XSE-XSW) )

C     COMPUTE THE TEMPERATURE GRADIENTS FOR VISCOUS CALCULATIONS

      DTDXW   = 2.*RVOLM2(ICELL)*( TWO*(YSW-YNW) + TN2*(YNW-YNO)
     1                           + TCO*(YNO-YSO) + TS1*(YSO-YSW) )
      DTDXN   = 2.*RVOLM2(ICELL)*( TW1*(YWO-YNW) + TNO*(YNW-YNE)
     1                           + TE2*(YNE-YEO) + TCO*(YEO-YWO) )
      DTDXE   = 2.*RVOLM2(ICELL)*( TCO*(YSO-YNO) + TN1*(YNO-YNE)
     1                           + TEO*(YNE-YSE) + TS2*(YSE-YSO) )
      DTDXS   = 2.*RVOLM2(ICELL)*( TW2*(YSW-YWO) + TCO*(YWO-YEO)
     1                           + TE1*(YEO-YSE) + TSO*(YSE-YSW) )

      DTDYW   =-2.*RVOLM2(ICELL)*( TWO*(XSW-XNW) + TN2*(XNW-XNO)
     1                           + TCO*(XNO-XSO) + TS1*(XSO-XSW) )
      DTDYN   =-2.*RVOLM2(ICELL)*( TW1*(XWO-XNW) + TNO*(XNW-XNE)
     1                           + TE2*(XNE-XEO) + TCO*(XEO-XWO) )
      DTDYE   =-2.*RVOLM2(ICELL)*( TCO*(XSO-XNO) + TN1*(XNO-XNE)
     1                           + TEO*(XNE-XSE) + TS2*(XSE-XSO) )
      DTDYS   =-2.*RVOLM2(ICELL)*( TW2*(XSW-XWO) + TCO*(XWO-XEO)
     1                           + TE1*(XEO-XSE) + TSO*(XSE-XSW) )

C     COMPUTE THE VISOCITY COEFFICIENT AS GIVEN BY THE POWER LAW
C     FOR VISCOUS CALCULATIONS
      AMSW    = TEMPG2(KSW)**OMEGE2
      AMSE    = TEMPG2(KSE)**OMEGE2
      AMNE    = TEMPG2(KNE)**OMEGE2
      AMNW    = TEMPG2(KNW)**OMEGE2

      AMSO    = 0.5*(AMSW+AMSE)
      AMEO    = 0.5*(AMSE+AMNE)
      AMNO    = 0.5*(AMNE+AMNW)
      AMWO    = 0.5*(AMNW+AMSW)
```

```
C          COMPUTE THE THERMAL CONDUCTIVITY AS GIVEN BY THE POWER LAW
C          TIMES THE GAMMA FACTOR FOR VISCOUS CALCULATIONS
           CNSO     = AMSO*GFACE2
           CNEO     = AMEO*GFACE2
           CNNO     = AMNO*GFACE2
           CNWO     = AMWO*GFACE2

C          COMPUTE THE VISOUS TERMS FOR MOMENTUM EQUATIONS

           AVISXX   = AMWO*(2.*DUDXW-DVDYW)*(YNW-YSW) +
     1                AMNO*(2.*DUDXN-DVDYN)*(YNE-YNW) +
     1                AMEO*(2.*DUDXE-DVDYE)*(YSE-YNE) +
     1                AMSO*(2.*DUDXS-DVDYS)*(YSW-YSE)
           AVISXY   = AMWO*(DUDYW+DVDXW)*(XNW-XSW) +
     1                AMNO*(DUDYN+DVDXN)*(XNE-XNW) +
     1                AMEO*(DUDYE+DVDXE)*(XSE-XNE) +
     1                AMSO*(DUDYS+DVDXS)*(XSW-XSE)

           AVISYX   = AMWO*(DUDYW+DVDXW)*(YNW-YSW) +
     1                AMNO*(DUDYN+DVDXN)*(YNE-YNW) +
     1                AMEO*(DUDYE+DVDXE)*(YSE-YNE) +
     1                AMSO*(DUDYS+DVDXS)*(YSW-YSE)

           AVISYY   = AMWO*(2.*DVDYW-DUDXW)*(XNW-XSW) +
     1                AMNO*(2.*DVDYN-DUDXN)*(XNE-XNW) +
     1                AMEO*(2.*DVDYE-DUDXE)*(XSE-XNE) +
     1                AMSO*(2.*DVDYS-DUDXS)*(XSW-XSE)

C          COMPUTE THE VISOUS TERMS FOR ENERGY EQUATIONS

           AENEX1   = AMWO*UWO*(2.*DUDXW-DVDYW)*(YNW-YSW) +
     1                AMNO*UNO*(2.*DUDXN-DVDYN)*(YNE-YNW) +
     1                AMEO*UEO*(2.*DUDXE-DVDYE)*(YSE-YNE) +
     1                AMSO*USO*(2.*DUDXS-DVDYS)*(YSW-YSE)

           AENEX2   = AMWO*VWO*(DUDYW+DVDXW)*(YNW-YSW) +
     1                AMNO*VNO*(DUDYN+DVDXN)*(YNE-YNW) +
     1                AMEO*VEO*(DUDYE+DVDXE)*(YSE-YNE) +
     1                AMSO*VSO*(DUDYS+DVDXS)*(YSW-YSE)

           AENEX3   = RPRNE2*( CNWO*DTDXW*(YNW-YSW) +
     1                         CNNO*DTDXN*(YNE-YNW) +
     1                         CNEO*DTDXE*(YSE-YNE) +
     1                         CNSO*DTDXS*(YSW-YSE) )

           AENEY1   = AMWO*UWO*(DUDYW+DVDXW)*(XNW-XSW) +
     1                AMNO*UNO*(DUDYN+DVDXN)*(XNE-XNW) +
     1                AMEO*UEO*(DUDYE+DVDXE)*(XSE-XNE) +
     1                AMSO*USO*(DUDYS+DVDXS)*(XSW-XSE)

           AENEY2   = AMWO*VWO*(2.*DVDYW-DUDXW)*(XNW-XSW) +
     1                AMNO*VNO*(2.*DVDYN-DUDXN)*(XNE-XNW) +
     1                AMEO*VEO*(2.*DVDYE-DUDXE)*(XSE-XNE) +
     1                AMSO*VSO*(2.*DVDYS-DUDXS)*(XSW-XSE)

           AENEY3   = RPRNE2*( CNWO*DTDYW*(XNW-XSW) +
     1                         CNNO*DTDYN*(XNE-XNW) +
```

```
     1                        CNEO*DTDYE*(XSE-XNE) +
     1                        CNSO*DTDYS*(XSW-XSE) )
      -
      -
          TFACTOR = -RREYE2*DTDVOL

          DVISC(1) = 0.
          DVISC(2) = TFACTOR*(2./3.*AVISXX - AVISXY)
          DVISC(3) = TFACTOR*(AVISYX - 2./3.*AVISYY)
          DVISC(4) = TFACTOR*((2./3.*AENEX1+AENEX2+AENEX3) -
     1                    (2./3.*AENEY2+AENEY1+AENEY3) )

C         COMPUTE THE VISCOUS TERMS PERTAINING TO SPECIES EQUATIONS
          DO 425 J = NEQBAS + 1, NEQNFL
             AYSW  = DPENG2(J,KSW)/DPENG2(1,KSW)
             AYSE  = DPENG2(J,KSE)/DPENG2(1,KSE)
             AYNE  = DPENG2(J,KNE)/DPENG2(1,KNE)
             AYNW  = DPENG2(J,KNW)/DPENG2(1,KNW)
             AYSO  = 0.5*(AYSW+AYSE)
             AYEO  = 0.5*(AYSE+AYNE)
             AYNO  = 0.5*(AYNE+AYNW)
             AYWO  = 0.5*(AYNW+AYSW)
             AYS1  = 0.5*(AYSW+AYSO)
             AYS2  = 0.5*(AYSO+AYSE)
             AYE1  = 0.5*(AYSE+AYEO)
             AYE2  = 0.5*(AYEO+AYNE)
             AYN1  = 0.5*(AYNE+AYNO)
             AYN2  = 0.5*(AYNO+AYNW)
             AYW1  = 0.5*(AYNW+AYWO)
             AYW2  = 0.5*(AYWO+AYSW)

             DADXW = 2.*RVOLM2(ICELL)*(AYWO*(YSW-YNW)
     1              +AYN2*(YNW-YNO) +AYCO*(YNO-YSO) +AYS1*(YSO-YSW))
             DADXN = 2.*RVOLM2(ICELL)*(AYW1*(YWO-YNW)
     1              +AYNO*(YNW-YNE) +AYE2*(YNE-YEO) +AYCO*(YEO-YWO))
             DADXE = 2.*RVOLM2(ICELL)*(AYCO*(YSO-YNO)
     1              +AYN1*(YNO-YNE) +AYEO*(YNE-YSE) +AYS2*(YSE-YSO))
             DADXS = 2.*RVOLM2(ICELL)*(AYW2*(YSW-YWO)
     1              +AYCO*(YWO-YEO) +AYE1*(YEO-YSE) +AYSO*(YSE-YSW))

             DADYW =-2.*RVOLM2(ICELL)*(AYWO*(XSW-XNW)
     1              +AYN2*(XNW-XNO) +AYCO*(XNO-XSO) +AYS1*(XSO-XSW))
             DADYN =-2.*RVOLM2(ICELL)*(AYW1*(XWO-XNW)
     1              +AYNO*(XNW-XNE) +AYE2*(XNE-XEO) +AYCO*(XEO-XWO))
             DADYE =-2.*RVOLM2(ICELL)*(AYCO*(XSO-XNO)
     1              +AYN1*(XNO-XNE) +AYEO*(XNE-XSE) +AYS2*(XSE-XSO))
             DADYS =-2.*RVOLM2(ICELL)*(AYW2*(XSW-XWO)
     1              +AYCO*(XWO-XEO) +AYE1*(XEO-XSE) +AYSO*(XSE-XSW))


             ADIFX   = AMWO*DADXW*(YNW-YSW) + AMNO*DADXN*(YNE-YNW) +
     1                 AMEO*DADXE*(YSE-YNE) + AMSO*DADXS*(YSW-YSE)
             ADIFY   = AMWO*DADYW*(XNW-XSW) + AMNO*DADYN*(XNE-XNW) +
     1                 AMEO*DADYE*(XSE-XNE) + AMSO*DADYS*(XSW-XSE)

             DVISC(J) = TFACTOR*RSCHE2*(ADIFX - ADIFY)

425       CONTINUE
```

```
C          --------------------
C          JACOBIAN CHANGE BLOCK
C          --------------------

C          COMPUTE CHANGES DUE TO JACOBIANS

CVD$       NOLSTVAL
           DO 440 J = 1, NEQNFL
              DFCELL = 0.
              DGCELL = 0.

              DO 430 K = 1, NEQNFL
                 DFCELL = DFCELL + FUJACO(J,K)*DUCELL(K)
                 DGCELL = DGCELL + GUJACO(J,K)*DUCELL(K)
430           CONTINUE

C             TRANSFORM THE JACOBIAN CHANGES (ONLY FU AND GU) AND
C             MULTIPLY WITH THEIR RESPECTIVE SCALINGS OF TIME

              TEMPF  = DFCELL
              DFCELL = DTDVOL*( TEMPF*DYNSM2(ICELL)
     1                                 -DGCELL*DXNSM2(ICELL))
              DGCELL = DTDVOL*(-TEMPF*DYEWM2(ICELL)
     1                                 +DGCELL*DXEWM2(ICELL))

C             ---------------
C             DIFFUSION  TERMS
C             ---------------

C             COMPUTE THE DIFFUSION TERMS FOR THE FOUR EDGES

              SIGGSW = SIGGE2(KSW)*DPENG2(J,KSW)
              SIGGSE = SIGGE2(KSE)*DPENG2(J,KSE)
              SIGGNE = SIGGE2(KNE)*DPENG2(J,KNE)
              SIGGNW = SIGGE2(KNW)*DPENG2(J,KNW)

C             COMPUTE THE DIFFUSION TERM FOR THE WHOLE CELL

              SIGCEL= 0.25*(SIGGSW + SIGGSE + SIGGNE + SIGGNW)
              SIGGSW = SIGCEL - SIGGSW
              SIGGSE = SIGCEL - SIGGSE
              SIGGNE = SIGCEL - SIGGNE
              SIGGNW = SIGCEL - SIGGNW
C
              SIGGSW = DSDIFF*SIGGSW
              SIGGSE = DSDIFF*SIGGSE
              SIGGNE = DSDIFF*SIGGNE
              SIGGNW = DSDIFF*SIGGNW

C             ---------------------
C             COMPUTATION OF CHANGES
C             ---------------------

C             FOCIT IS DUCELL; FIND SOCIT AND CORNER CHANGES

              SOCITSW  = - DFCELL - DGCELL
```

```
               SOCITNW  = - DFCELL + DGCELL
               SOCITNE  = + DFCELL + DGCELL
               SOCITSE  = + DFCELL - DGCELL

               DELSW(J) = 0.25*( DUCELL(J) + SOCITSW + SIGGSW )
               DELNW(J) = 0.25*( DUCELL(J) + SOCITNW + SIGGNW )
               DELNE(J) = 0.25*( DUCELL(J) + SOCITNE + SIGGNE )
               DELSE(J) = 0.25*( DUCELL(J) + SOCITSE + SIGGSE )

440         CONTINUE
C
C           DO IMPLICIT SOURCE TERMS

            CALL PTIMP2 (KSW, ICELL, DELSW)
            CALL PTIMP2 (KSE, ICELL, DELSE)
            CALL PTIMP2 (KNW, ICELL, DELNW)
            CALL PTIMP2 (KNE, ICELL, DELNE)

C           ----------------------
C           DISTRIBUTION OF CHANGES
C           ----------------------

C           DISTRIBUTE CONVECTIVE AND DIFFUSIVE CHANGES

CVD$        NOLSTVAL
            DO 450 J = 1, NEQNFL
               CHNGE2(J,KSW) = CHNGE2(J,KSW) + DELSW(J)
               CHNGE2(J,KNW) = CHNGE2(J,KNW) + DELNW(J)
               CHNGE2(J,KSE) = CHNGE2(J,KSE) + DELSE(J)
               CHNGE2(J,KNE) = CHNGE2(J,KNE) + DELNE(J)
450         CONTINUE

560      CONTINUE
C
         RETURN
C
C        USE EXPLICIT SOURCE TERMS, KEEPING THE CHEMISTRY FROZEN
C        STEP THROUGH EACH CELL AT THIS LEVEL
C
CVD$     NOLSTVAL
610      DO 710 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C           ----------------------
C           CELL/NODE DETERMINATION
C           ----------------------

C           FIND THE CELL TO BE INTEGRATED

            ICELL = ICELTI(JCELL)

C           SET UP NODE POINTERS FOR THIS CELL

            KSW = ICELG2( 2,ICELL)
            KSE = ICELG2( 4,ICELL)
            KNE = ICELG2( 6,ICELL)
            KNW = ICELG2( 8,ICELL)
```

```
C          --------
C          GEOMETRY
C          --------
C
C          GEOMETRY OF ALL CELL CORNERS

           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)

           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)

           XNE = GEOMG2(1,KNE)
           YNE = GEOMG2(2,KNE)

           XNW = GEOMG2(1,KNW)
           YNW = GEOMG2(2,KNW)
C
C          THE RATIO DELTA-t TO CELL VOLUME
           DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

C          COMPUTE THE AVERAGE COEFFICIENT FOR DIFFUSION

           DSDIFF = 0.5*PERIM2(ICELL)*DTDVOL

C          -------------
C          FACIAL VALUES
C          -------------
C
C          COMPUTE THE DEPENDENT VARIABLES AT THE FACES

           PRESSS = 0.5*( PRESG2(KSW) + PRESG2(KSE) )
           PRESSE = 0.5*( PRESG2(KSE) + PRESG2(KNE) )
           PRESSN = 0.5*( PRESG2(KNW) + PRESG2(KNE) )
           PRESSW = 0.5*( PRESG2(KSW) + PRESG2(KNW) )

CVD$       NOLSTVAL
           DO 620 IQ = 1, 4
              DPENFA(IQ,1) = 0.5*( DPENG2(IQ,KSW) + DPENG2(IQ,KSE) )
              DPENFA(IQ,2) = 0.5*( DPENG2(IQ,KSE) + DPENG2(IQ,KNE) )
              DPENFA(IQ,3) = 0.5*( DPENG2(IQ,KNE) + DPENG2(IQ,KNW) )
              DPENFA(IQ,4) = 0.5*( DPENG2(IQ,KNW) + DPENG2(IQ,KSW) )
620        CONTINUE

           UCOMPS = DPENFA(2,1)/DPENFA(1,1)
           VCOMPS = DPENFA(3,1)/DPENFA(1,1)
           UCOMPE = DPENFA(2,2)/DPENFA(1,2)
           VCOMPE = DPENFA(3,2)/DPENFA(1,2)
           UCOMPN = DPENFA(2,3)/DPENFA(1,3)
           VCOMPN = DPENFA(3,3)/DPENFA(1,3)
           UCOMPW = DPENFA(2,4)/DPENFA(1,4)
           VCOMPW = DPENFA(3,4)/DPENFA(1,4)

C          ----------
C          FLUX TERMS
C          ----------
C          SOUTH
```

```
            BIGFS(1) = DPENFA(2,1)
            BIGFS(2) = DPENFA(2,1)*UCOMPS + PRESSS
            BIGFS(3) = DPENFA(2,1)*VCOMPS
            BIGFS(4) = UCOMPS*(DPENFA(4,1) + PRESSS)

            BIGGS(1) = DPENFA(3,1)
            BIGGS(2) = BIGFS(3)
            BIGGS(3) = DPENFA(3,1)*VCOMPS + PRESSS
            BIGGS(4) = VCOMPS*(DPENFA(4,1) + PRESSS)

C           EAST

            BIGFE(1) = DPENFA(2,2)
            BIGFE(2) = DPENFA(2,2)*UCOMPE + PRESSE
            BIGFE(3) = DPENFA(2,2)*VCOMPE
            BIGFE(4) = UCOMPE*(DPENFA(4,2) + PRESSE)
            BIGGE(1) = DPENFA(3,2)
            BIGGE(2) = BIGFE(3)
            BIGGE(3) = DPENFA(3,2)*VCOMPE + PRESSE
            BIGGE(4) = VCOMPE*(DPENFA(4,2) + PRESSE)

C           NORTH

            BIGFN(1) = DPENFA(2,3)
            BIGFN(2) = DPENFA(2,3)*UCOMPN + PRESSN
            BIGFN(3) = DPENFA(2,3)*VCOMPN
            BIGFN(4) = UCOMPN*(DPENFA(4,3) + PRESSN)
            BIGGN(1) = DPENFA(3,3)
            BIGGN(2) = BIGFN(3)
            BIGGN(3) = DPENFA(3,3)*VCOMPN + PRESSN
            BIGGN(4) = VCOMPN*(DPENFA(4,3) + PRESSN)

C           WEST

            BIGFW(1) = DPENFA(2,4)
            BIGFW(2) = DPENFA(2,4)*UCOMPW + PRESSW
            BIGFW(3) = DPENFA(2,4)*VCOMPW
            BIGFW(4) = UCOMPW*(DPENFA(4,4) + PRESSW)
            BIGGW(1) = DPENFA(3,4)
            BIGGW(2) = BIGFW(3)
            BIGGW(3) = DPENFA(3,4)*VCOMPW + PRESSW
            BIGGW(4) = VCOMPW*(DPENFA(4,4) + PRESSW)

C           --------------
C           JACOBIAN TERMS
C           --------------
C
C           DEFINE DEPENDENT VARIABLES AT THE CENTER OF THE CELL

CVD$        NOLSTVAL
            DO 630 IQ = 1, 4
               DPENJA(IQ) = 0.25*( DPENFA(IQ,1) + DPENFA(IQ,2) +
     1                             DPENFA(IQ,3) + DPENFA(IQ,4) )
630         CONTINUE
C
C           NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
```

```
C          UCOMPC, VCOMPC, GAMAPR, YSPEPR ETC. AND GET THE SOURCE TERMS
C          FOR THE CELL
C           :
           UCOMPC = DPENJA(2)/DPENJA(1)
           VCOMPC = DPENJA(3)/DPENJA(1)
           U2     = UCOMPC*UCOMPC
           V2     = VCOMPC*VCOMPC
           BEPSPR = DPENJA(4)
           BEU    = BEPSPR/DPENJA(1)
           VELO2U = U2 + V2
C
C          COMPUTE THE DIMENSIONAL QUANTITIES
C
           BE     = FMREFL*BEU
           VELO2  = FMREFL*VELO2U

C          COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

           SUMY = 0.
           DO 640 IS = 1, NEQSCH
              JS         = NEQBAS + IS
              YSPEPR(IS) = DPENJA(JS)/DPENJA(1)
              SUMY       = SUMY + YSPEPR(IS)
640        CONTINUE

           YNEXT            = 1. - SUMY - YNRTCH
           IF (YNEXT .LT. 0.) YNEXT = 0.
           YSPEPR(NEQSCH+1) = YNEXT
C
           SYSHFS = 0.
           SYSCPS = 0.
           SYSBMS = 0.
           BIGAM  = 0.
C
C          COMPUTE THE TEMPERATURE IN DEGREE K AND ALSO
C
           DO 650 IS = 1, NSPECH
              SYSHFS     = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
              SYSCPS     = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
              SYSBMS     = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
              BIGAM      = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
650        CONTINUE
C
C          COMPUTE TEMPERATURE IN DEGREE K AND SOME RELATED QUANTITIES

           BIGBM   = SYSCPS - UGASFL*SYSBMS
           BIGCM   = BE - 0.5*VELO2 - SYSHFS + TREFCH*SYSCPS
     1                          + 0.5*TREFCH*TREFCH*BIGAM
           IF (BIGAM .LT. 1.E-10) THEN
              TEMP   = BIGCM/BIGBM
           ELSE
              DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
              TEMP   = ( SQRT(DISCRI)-BIGBM )/BIGAM
           ENDIF

           BIGAMT = BIGAM *TEMP
           SYSCVS = BIGBM + BIGAMT
```

```
      GAMAPR = (SYSCPS+BIGAMT)/SYSCVS
C
C     NORMALIZE THE TEMPERATURE
C
      TEMPU = TEMP/TREFFL
C
C     COMPUTE THE DIMENSIONLESS PRESSURE
C
      PRESPR = DPENJA(1)*TEMPU*AMWTFL*SYSBMS

      GM1    = GAMAPR - 1.
      GM3    = GM1 - 2.
      PAEBR  = (DPENJA(4)+PRESPR)/DPENJA(1)

      FUJACO(1,2) = 1.
      GUJACO(1,3) = 1.
C
      GUJACO(2,1) = -UCOMPC*VCOMPC
      GUJACO(2,2) =  VCOMPC
      GUJACO(2,3) =  UCOMPC
C
      FUJACO(3,1) = GUJACO(2,1)
      FUJACO(3,2) = GUJACO(2,2)
      FUJACO(3,3) = GUJACO(2,3)
C
      FUJACO(2,1) = 0.5*(GM3*U2 + GM1*V2)
      FUJACO(2,2) = -GM3*UCOMPC
      FUJACO(2,3) = -GM1*VCOMPC
      FUJACO(2,4) =  GM1
C
      GUJACO(3,1) = FUJACO(2,1) - V2 + U2
      GUJACO(3,2) = FUJACO(2,2) - 2.*UCOMPC
      GUJACO(3,3) = FUJACO(2,3) - 2.*VCOMPC
      GUJACO(3,4) = FUJACO(2,4)
C
      FUJACO(4,1) = UCOMPC*(FUJACO(2,1) + U2 - PAEBR)
      FUJACO(4,2) = PAEBR - 2.*U2 + UCOMPC*FUJACO(2,2)
      FUJACO(4,3) = UCOMPC*FUJACO(2,3)
      FUJACO(4,4) = UCOMPC*(FUJACO(2,4) + 1.)
C
      GUJACO(4,1) = VCOMPC*(FUJACO(2,1) + U2 - PAEBR)
      GUJACO(4,2) = VCOMPC*(FUJACO(2,2) - 2.*UCOMPC)
      GUJACO(4,3) = VCOMPC*FUJACO(2,3) + PAEBR
      GUJACO(4,4) = VCOMPC*(FUJACO(2,4) + 1.)
C
C     ------------------------------
C     FIRST ORDER CELL CHANGE DUCELL
C     ------------------------------

C     CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE

CVD$  NOLSTVAL
      DO 660 J = 1, 4
         DUCELL(J) = DTDVOL*(
     1                BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1                BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1                BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
```

```
     1                    BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )
660        CONTINUE

C          COMPUTE THE DISTANCES FOR THE CELL UNDER CONSIDERATION
           XSO      = 0.5*(XSW+XSE)
           XEO      = 0.5*(XSE+XNE)
           XNO      = 0.5*(XNE+XNW)
           XWO      = 0.5*(XNW+XSW)

           YSO      = 0.5*(YSW+YSE)
           YEO      = 0.5*(YSE+YNE)
           YNO      = 0.5*(YNE+YNW)
           YWO      = 0.5*(YNW+YSW)

C          COMPUTE THE VELOCITY COMPONENTS AT SPECIAL POINTERS FOR
C          VISCOUS CALCULATIONS
           USW      = DPENG2(2,KSW)/DPENG2(1,KSW)
           USE      = DPENG2(2,KSE)/DPENG2(1,KSE)
           UNE      = DPENG2(2,KNE)/DPENG2(1,KNE)
           UNW      = DPENG2(2,KNW)/DPENG2(1,KNW)

           USO      = 0.5*(USW+USE)
           UEO      = 0.5*(USE+UNE)
           UNO      = 0.5*(UNE+UNW)
           UWO      = 0.5*(UNW+USW)
           UCO      = 0.25*(USW+USE+UNE+UNW)

           US1      = 0.5*(USW+USO)
           US2      = 0.5*(USO+USE)
           UE1      = 0.5*(USE+UEO)
           UE2      = 0.5*(UEO+UNE)
           UN1      = 0.5*(UNE+UNO)
           UN2      = 0.5*(UNO+UNW)
           UW1      = 0.5*(UNW+UWO)
           UW2      = 0.5*(UWO+USW)

           VSW      = DPENG2(3,KSW)/DPENG2(1,KSW)
           VSE      = DPENG2(3,KSE)/DPENG2(1,KSE)
           VNE      = DPENG2(3,KNE)/DPENG2(1,KNE)
           VNW      = DPENG2(3,KNW)/DPENG2(1,KNW)

           VSO      = 0.5*(VSW+VSE)
           VEO      = 0.5*(VSE+VNE)
           VNO      = 0.5*(VNE+VNW)
           VWO      = 0.5*(VNW+VSW)
           VCO      = 0.25*(VSW+VSE+VNE+VNW)

           VS1      = 0.5*(VSW+VSO)
           VS2      = 0.5*(VSO+VSE)
           VE1      = 0.5*(VSE+VEO)
           VE2      = 0.5*(VEO+VNE)
           VN1      = 0.5*(VNE+VNO)
           VN2      = 0.5*(VNO+VNW)
           VW1      = 0.5*(VNW+VWO)
           VW2      = 0.5*(VWO+VSW)

C          COMPUTE THE TEMPERATURE FOR VISCOUS CALCULATIONS
```

```
      TSW     = TEMPG2(KSW)
      TSE     = TEMPG2(KSE)
      TNE     = TEMPG2(KNE)
      TNW     = TEMPG2(KNW)

      TSO     = 0.5*(TSW+TSE)
      TEO     = 0.5*(TSE+TNE)
      TNO     = 0.5*(TNE+TNW)
      TWO     = 0.5*(TNW+TSW)
      TCO     = 0.25*(TSW+TSE+TNE+TNW)

      TS1     = 0.5*(TSW+TSO)
      TS2     = 0.5*(TSO+TSE)
      TE1     = 0.5*(TSE+TEO)
      TE2     = 0.5*(TEO+TNE)
      TN1     = 0.5*(TNE+TNO)
      TN2     = 0.5*(TNO+TNW)
      TW1     = 0.5*(TNW+TWO)
      TW2     = 0.5*(TWO+TSW)

C     COMPUTE THE VELOCITY GRADIENTS FOR VISCOUS CALCULATIONS

      DUDXW   = 2.*RVOLM2(ICELL)*( UWO*(YSW-YNW) + UN2*(YNW-YNO)
     1                           + UCO*(YNO-YSO) + US1*(YSO-YSW) )
      DUDXN   = 2.*RVOLM2(ICELL)*( UW1*(YWO-YNW) + UNO*(YNW-YNE)
     1                           + UE2*(YNE-YEO) + UCO*(YEO-YWO) )
      DUDXE   = 2.*RVOLM2(ICELL)*( UCO*(YSO-YNO) + UN1*(YNO-YNE)
     1                           + UEO*(YNE-YSE) + US2*(YSE-YSO) )
      DUDXS   = 2.*RVOLM2(ICELL)*( UW2*(YSW-YWO) + UCO*(YWO-YEO)
     1                           + UE1*(YEO-YSE) + USO*(YSE-YSW) )

      DUDYW   =-2.*RVOLM2(ICELL)*( UWO*(XSW-XNW) + UN2*(XNW-XNO)
     1                           + UCO*(XNO-XSO) + US1*(XSO-XSW) )
      DUDYN   =-2.*RVOLM2(ICELL)*( UW1*(XWO-XNW) + UNO*(XNW-XNE)
     1                           + UE2*(XNE-XEO) + UCO*(XEO-XWO) )
      DUDYE   =-2.*RVOLM2(ICELL)*( UCO*(XSO-XNO) + UN1*(XNO-XNE)
     1                           + UEO*(XNE-XSE) + US2*(XSE-XSO) )
      DUDYS   =-2.*RVOLM2(ICELL)*( UW2*(XSW-XWO) + UCO*(XWO-XEO)
     1                           + UE1*(XEO-XSE) + USO*(XSE-XSW) )

      DVDXW   = 2.*RVOLM2(ICELL)*( VWO*(YSW-YNW) + VN2*(YNW-YNO)
     1                           + VCO*(YNO-YSO) + VS1*(YSO-YSW) )
      DVDXN   = 2.*RVOLM2(ICELL)*( VW1*(YWO-YNW) + VNO*(YNW-YNE)
     1                           + VE2*(YNE-YEO) + VCO*(YEO-YWO) )
      DVDXE   = 2.*RVOLM2(ICELL)*( VCO*(YSO-YNO) + VN1*(YNO-YNE)
     1                           + VEO*(YNE-YSE) + VS2*(YSE-YSO) )
      DVDXS   = 2.*RVOLM2(ICELL)*( VW2*(YSW-YWO) + VCO*(YWO-YEO)
     1                           + VE1*(YEO-YSE) + VSO*(YSE-YSW) )

      DVDYW   =-2.*RVOLM2(ICELL)*( VWO*(XSW-XNW) + VN2*(XNW-XNO)
     1                           + VCO*(XNO-XSO) + VS1*(XSO-XSW) )
      DVDYN   =-2.*RVOLM2(ICELL)*( VW1*(XWO-XNW) + VNO*(XNW-XNE)
     1                           + VE2*(XNE-XEO) + VCO*(XEO-XWO) )
      DVDYE   =-2.*RVOLM2(ICELL)*( VCO*(XSO-XNO) + VN1*(XNO-XNE)
     1                           + VEO*(XNE-XSE) + VS2*(XSE-XSO) )
      DVDYS   =-2.*RVOLM2(ICELL)*( VW2*(XSW-XWO) + VCO*(XWO-XEO)
     1                           + VE1*(XEO-XSE) + VSO*(XSE-XSW) )
```

```
C         COMPUTE THE TEMPERATURE GRADIENTS FOR VISCOUS CALCULATIONS
          -
          DTDXW   = 2.*RVOLM2(ICELL)*( TWO*(YSW-YNW) + TN2*(YNW-YNO)
     1                               + TCO*(YNO-YSO) + TS1*(YSO-YSW) )
          DTDXN   = 2.*RVOLM2(ICELL)*( TW1*(YWO-YNW) + TNO*(YNW-YNE)
     1                               + TE2*(YNE-YEO) + TCO*(YEO-YWO) )
          DTDXE   = 2.*RVOLM2(ICELL)*( TCO*(YSO-YNO) + TN1*(YNO-YNE)
     1                               + TEO*(YNE-YSE) + TS2*(YSE-YSO) )
          DTDXS   = 2.*RVOLM2(ICELL)*( TW2*(YSW-YWO) + TCO*(YWO-YEO)
     1                               + TE1*(YEO-YSE) + TSO*(YSE-YSW) )

          DTDYW   =-2.*RVOLM2(ICELL)*( TWO*(XSW-XNW) + TN2*(XNW-XNO)
     1                               + TCO*(XNO-XSO) + TS1*(XSO-XSW) )
          DTDYN   =-2.*RVOLM2(ICELL)*( TW1*(XWO-XNW) + TNO*(XNW-XNE)
     1                               + TE2*(XNE-XEO) + TCO*(XEO-XWO) )
          DTDYE   =-2.*RVOLM2(ICELL)*( TCO*(XSO-XNO) + TN1*(XNO-XNE)
     1                               + TEO*(XNE-XSE) + TS2*(XSE-XSO) )
          DTDYS   =-2.*RVOLM2(ICELL)*( TW2*(XSW-XWO) + TCO*(XWO-XEO)
     1                               + TE1*(XEO-XSE) + TSO*(XSE-XSW) )

C         COMPUTE THE VISOCITY COEFFICIENT AS GIVEN BY THE POWER LAW
C         FOR VISCOUS CALCULATIONS
          AMSW    = TEMPG2(KSW)**OMEGE2
          AMSE    = TEMPG2(KSE)**OMEGE2
          AMNE    = TEMPG2(KNE)**OMEGE2
          AMNW    = TEMPG2(KNW)**OMEGE2

          AMSO    = 0.5*(AMSW+AMSE)
          AMEO    = 0.5*(AMSE+AMNE)
          AMNO    = 0.5*(AMNE+AMNW)
          AMWO    = 0.5*(AMNW+AMSW)

C         COMPUTE THE THERMAL CONDUCTIVITY AS GIVEN BY THE POWER LAW
C         TIMES THE GAMMA FACTOR FOR VISCOUS CALCULATIONS
          CNSO    = AMSO*GFACE2
          CNEO    = AMEO*GFACE2
          CNNO    = AMNO*GFACE2
          CNWO    = AMWO*GFACE2

C         COMPUTE THE VISOUS TERMS FOR MOMENTUM EQUATIONS

          AVISXX  = AMWO*(2.*DUDXW-DVDYW)*(YNW-YSW) +
     1              AMNO*(2.*DUDXN-DVDYN)*(YNE-YNW) +
     1              AMEO*(2.*DUDXE-DVDYE)*(YSE-YNE) +
     1              AMSO*(2.*DUDXS-DVDYS)*(YSW-YSE)
          AVISXY  = AMWO*(DUDYW+DVDXW)*(XNW-XSW) +
     1              AMNO*(DUDYN+DVDXN)*(XNE-XNW) +
     1              AMEO*(DUDYE+DVDXE)*(XSE-XNE) +
     1              AMSO*(DUDYS+DVDXS)*(XSW-XSE)

          AVISYX  = AMWO*(DUDYW+DVDXW)*(YNW-YSW) +
     1              AMNO*(DUDYN+DVDXN)*(YNE-YNW) +
     1              AMEO*(DUDYE+DVDXE)*(YSE-YNE) +
     1              AMSO*(DUDYS+DVDXS)*(YSW-YSE)

          AVISYY  = AMWO*(2.*DVDYW-DUDXW)*(XNW-XSW) +
```

```
     1                AMNO*(2.*DVDYN-DUDXN)*(XNE-XNW) +
     1      -         AMEO*(2.*DVDYE-DUDXE)*(XSE-XNE) +
     1      -         AMSO*(2.*DVDYS-DUDXS)*(XSW-XSE)

C          COMPUTE THE VISOUS TERMS FOR ENERGY EQUATIONS

           AENEX1   = AMWO*UWO*(2.*DUDXW-DVDYW)*(YNW-YSW) +
     1                AMNO*UNO*(2.*DUDXN-DVDYN)*(YNE-YNW) +
     1                AMEO*UEO*(2.*DUDXE-DVDYE)*(YSE-YNE) +
     1                AMSO*USO*(2.*DUDXS-DVDYS)*(YSW-YSE)

           AENEX2   = AMWO*VWO*(DUDYW+DVDXW)*(YNW-YSW) +
     1                AMNO*VNO*(DUDYN+DVDXN)*(YNE-YNW) +
     1                AMEO*VEO*(DUDYE+DVDXE)*(YSE-YNE) +
     1                AMSO*VSO*(DUDYS+DVDXS)*(YSW-YSE)

           AENEX3   = RPRNE2*( CNWO*DTDXW*(YNW-YSW) +
     1                         CNNO*DTDXN*(YNE-YNW) +
     1                         CNEO*DTDXE*(YSE-YNE) +
     1                         CNSO*DTDXS*(YSW-YSE) )

           AENEY1   = AMWO*UWO*(DUDYW+DVDXW)*(XNW-XSW) +
     1                AMNO*UNO*(DUDYN+DVDXN)*(XNE-XNW) +
     1                AMEO*UEO*(DUDYE+DVDXE)*(XSE-XNE) +
     1                AMSO*USO*(DUDYS+DVDXS)*(XSW-XSE)

           AENEY2   = AMWO*VWO*(2.*DVDYW-DUDXW)*(XNW-XSW) +
     1                AMNO*VNO*(2.*DVDYN-DUDXN)*(XNE-XNW) +
     1                AMEO*VEO*(2.*DVDYE-DUDXE)*(XSE-XNE) +
     1                AMSO*VSO*(2.*DVDYS-DUDXS)*(XSW-XSE)

           AENEY3   = RPRNE2*( CNWO*DTDYW*(XNW-XSW) +
     1                         CNNO*DTDYN*(XNE-XNW) +
     1                         CNEO*DTDYE*(XSE-XNE) +
     1                         CNSO*DTDYS*(XSW-XSE) )

           TFACTOR = -RREYE2*DTDVOL

           DVISC(1) = 0.
           DVISC(2) = TFACTOR*(2./3.*AVISXX - AVISXY)
           DVISC(3) = TFACTOR*(AVISYX - 2./3.*AVISYY)
           DVISC(4) = TFACTOR*((2./3.*AENEX1+AENEX2+AENEX3) -
     1                         (2./3.*AENEY2+AENEY1+AENEY3) )

C          ---------------------
C          JACOBIAN CHANGE BLOCK
C          ---------------------

C          COMPUTE CHANGES DUE TO JACOBIANS

CVD$       NOLSTVAL
           DO 680 J = 1, 4
              DFCELL = 0.
              DGCELL = 0.

              DO 670 K = 1, 4
                 DFCELL = DFCELL + FUJACO(J,K)*DUCELL(K)
```

```
                DGCELL = DGCELL + GUJACO(J,K)*DUCELL(K)
670           CONTINUE
              -
C               TRANSFORM THE JACOBIAN CHANGES (ONLY FU AND GU) AND
C               MULTIPLY WITH THEIR RESPECTIVE SCALINGS OF TIME

                TEMPF  = DFCELL
                DFCELL = DTDVOL*( TEMPF*DYNSM2(ICELL)
     1                                -DGCELL*DXNSM2(ICELL))
                DGCELL = DTDVOL*(-TEMPF*DYEWM2(ICELL)
     1                                +DGCELL*DXEWM2(ICELL))

C               ---------------
C               DIFFUSION  TERMS
C               ---------------

C               COMPUTE THE DIFFUSION TERMS FOR THE FOUR EDGES

                SIGGSW = SIGGE2(KSW)*DPENG2(J,KSW)
                SIGGSE = SIGGE2(KSE)*DPENG2(J,KSE)
                SIGGNE = SIGGE2(KNE)*DPENG2(J,KNE)
                SIGGNW = SIGGE2(KNW)*DPENG2(J,KNW)

C               COMPUTE THE DIFFUSION TERM FOR THE WHOLE CELL

                SIGCEL= 0.25*(SIGGSW + SIGGSE + SIGGNE + SIGGNW)
                SIGGSW = SIGCEL - SIGGSW
                SIGGSE = SIGCEL - SIGGSE
                SIGGNE = SIGCEL - SIGGNE
                SIGGNW = SIGCEL - SIGGNW
C
                SIGGSW = DSDIFF*SIGGSW
                SIGGSE = DSDIFF*SIGGSE
                SIGGNE = DSDIFF*SIGGNE
                SIGGNW = DSDIFF*SIGGNW

C               ---------------------
C               COMPUTATION OF CHANGES
C               ---------------------

C               FOCIT IS DUCELL; FIND SOCIT AND CORNER CHANGES

                SOCITSW  = - DFCELL - DGCELL
                SOCITNW  = - DFCELL + DGCELL
                SOCITNE  = + DFCELL + DGCELL
                SOCITSE  = + DFCELL - DGCELL

                DELSW(J) = 0.25*( DUCELL(J) + SOCITSW + SIGGSW )
                DELNW(J) = 0.25*( DUCELL(J) + SOCITNW + SIGGNW )
                DELNE(J) = 0.25*( DUCELL(J) + SOCITNE + SIGGNE )
                DELSE(J) = 0.25*( DUCELL(J) + SOCITSE + SIGGSE )

680           CONTINUE

              DO 690 J = 5, NEQNFL
                 DELSW(J) = DELSW(1)*YSPEPR(J-4)
                 DELNW(J) = DELNW(1)*YSPEPR(J-4)
```

```
               DELNE(J) = DELNE(1)*YSPEPR(J-4)
              _ DELNW(J) = DELNW(1)*YSPEPR(J-4)
690         CONTINUE

C           ----------------------
C           DISTRIBUTION OF CHANGES
C           ----------------------

C           DISTRIBUTE CONVECTIVE AND DIFFUSIVE CHANGES

CVD$        NOLSTVAL
            DO 700 J = 1, NEQNFL
               CHNGE2(J,KSW) = CHNGE2(J,KSW) + DELSW(J)
               CHNGE2(J,KNW) = CHNGE2(J,KNW) + DELNW(J)
               CHNGE2(J,KSE) = CHNGE2(J,KSE) + DELSE(J)
               CHNGE2(J,KNE) = CHNGE2(J,KNE) + DELNE(J)
700         CONTINUE

710      CONTINUE
C
         RETURN
         END
```

E2SOUU

```
         SUBROUTINE E2SOUR
C                  E2SOUU

         INCLUDE 'PRECIS.INC'
         INCLUDE 'PARMV2.INC'
         INCLUDE 'CHCOMN.INC'
         INCLUDE 'E2COMN.INC'
         INCLUDE 'FLCOMN.INC'
         INCLUDE 'JACOMN.INC'
         INCLUDE 'PRCOMN.INC'
         DIMENSION WREACT(MREACH)
         DOUBLE PRECISION PROD1, PROD2, CONCEN(MSPECH)

C*********************************************************************

C        THIS FUNCTION COMPUTES THE SOURCE TERMS SO THAT THE JACOBIAN
C        TERMS COULD BE COMPUTED.  IT ALSO COMPUTES SOME FLUX TERMS, ONLY
C        TWO FLUX TERMS, F2, F4 AND G3, G4 ARE NEEDED FOR NUMERICAL
C        COMPUTATION OF FLUX JACOBIANS, WHEREAS ALL THE SOURCE TERMS ARE
C        NEEDED FOR SOURCE JACOBIANS.  THE TEMPORALLY VARYING VARIABLES ARE
C        STORED IN THE JA COMMON VARIABLES, I.E.,
C                     DPENJA(J) = DPENG1(J,INODE)
C        FOR THE GIVEN NODE INODE.
C        THE DIFFERENCE BETWEEN THIS ROUTINE AND E2FLUX IS THAT IN THAT
C        ROUTINE WE DO NOT CONSIDER VARIATIONS W.R.T. STATE VARIABLES.

C*********************************************************************
C
```

```
      RHO    = DPENJA(1)
      UCOMP  = DPENJA(2)/DPENJA(1)
      VCOMP  = DPENJA(3)/DPENJA(1)
      BEPS = DPENJA(4)
      BEU    = BEPS/RHO
      VELO2U = UCOMP*UCOMP + VCOMP*VCOMP
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BEU
      VELO2  = FMREFL*VELO2U
      RHOD   = RHO*RHORFL

C     COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

      SUMY = 0.

      DO 10 IS = 1, NEQSCH
          JS         = NEQBAS + IS
          YSPEPR(IS) = DPENJA(JS)/DPENJA(1)
          SUMY       = SUMY + YSPEPR(IS)
10      CONTINUE

      YNEXT            = 1. - SUMY - YNRTCH
      IF (YNEXT .LT. 0.) YNEXT = 0.
      YSPEPR(NEQSCH+1) = YNEXT

      SYSHFS = 0.
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
C     COMPUTE THE TEMPERATURE IN DEGREE K AND ALSO
C     COMPUTE THE CONCENTRATIONS OF ALL THE SPECIES IN KMOL/(M**3)

      DO 20 IS = 1, NSPECH
         SYSHFS     = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
         SYSCPS     = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
         SYSBMS     = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
         BIGAM      = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
         CONCEN(IS) = RHOD*YSPEPR(IS)*RAMWCH(IS)
         BIGWJA(IS) = 0.
20    CONTINUE
C
C     COMPUTE TEMPERATURE IN DEGREE K AND SOME RELATED QUANTITIES
C
      BIGBM   = SYSCPS - UGASFL*SYSBMS
      BIGCM   = BE - 0.5*VELO2 - SYSHFS + TREFCH*SYSCPS
     1                        + 0.5*TREFCH*TREFCH*BIGAM
      IF (BIGAM .LT. 1.E-10) THEN
         TEMP   = BIGCM/BIGBM
      ELSE
         DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
         TEMP   = ( SQRT(DISCRI)-BIGBM )/BIGAM
      ENDIF

      ALOGT  = LOG(ABS(TEMP))
```

```
      RTEMP  = 1./TEMP
C
C     NORMALIZE THE TEMPERATURE
C
      TEMPU = TEMP/TREFFL
C
C     COMPUTE THE DIMENSIONLESS PRESSURE
C
      PRESS = RHO*TEMPU*AMWTFL*SYSBMS

C     COMPUTE THE FLUX VARIABLES

      BGF2JA = DPENJA(2)*UCOMP + PRESS

C     BY-PASS THE REACTION CALCULATIONS IF TEMPERATURE IS LESS THAN
C     TRIGGER TEMPERATURE

      IF (TEMP .LT. TRIGCH) RETURN
      RECWDR = 1./WDREFL
C
C     CORRECT THE RATE COEFFICIENTS FOR ROGERS AND CHINITZ MODEL
C
      IF (KROGER .EQ. 1) THEN
         IF (YSPEPR(3) .LE. 0.) RETURN
         PHI       =  YSPEPR(3)*34.048/(1.-YSPEPR(3))
         IF (PHI .LT. 0.1 ) PHI = 0.1
         IF (PHI .GT. 2.0 ) PHI = 2.0
         RPHI      =  1./PHI
         TENLOG    =  LOG(10.)
         A1PHI     =  8.917*PHI + 31.433*RPHI - 28.95
         A2PHI     = -0.833*PHI +  1.333*RPHI +  2.00
         PREFCH(1) =  LOG(A1PHI) + 44.*TENLOG
         PREFCH(2) =  LOG(A2PHI) + 58.*TENLOG
         PREBCH(1) =  PREFCH(1) - PREECH(1)
         PREBCH(2) =  PREFCH(2) - PREECH(2)

C        USE THE FIRST REACTION AS EQUILIBRIUM REACTION, IF THE
C        CONCENTRATIONS ARE FAR AWAY FROM EQUILIBRIUM

         AKEQ   = 117.31948*EXP(-8992./TEMP)
         YOHEQ  = SQRT(YSPEPR(3)*YSPEPR(1)*AKEQ)
         DELTAY = YOHEQ-YSPEPR(2)

         IF ( DELTAY    .GT. 0.01*YMAXCH(2) .AND.
     1        YSPEPR(4) .LT. 0.50*YMAXCH(4)) THEN
            DELTAY    = 0.5*DELTAY*RAMWCH(2)
            YO2EQ     = YSPEPR(1) - AMWTCH(1)*DELTAY
            YH2EQ     = YSPEPR(3) - AMWTCH(3)*DELTAY
            CONCEN(1) = RHOD*YO2EQ*RAMWCH(1)
            CONCEN(2) = RHOD*YOHEQ*RAMWCH(2)
            CONCEN(3) = RHOD*YH2EQ*RAMWCH(3)
         ENDIF
C
C        REACTION # 1
C
         ALNKFR    = PREFCH(1) + EXPFCH(1)*ALOGT - ENEFCH(1)*RTEMP
         ALNKBR    = PREBCH(1) + EXPBCH(1)*ALOGT - ENEBCH(1)*RTEMP
```

```
            ALNKFR    = 0.5*ALNKFR
            ALNKBR    = 0.5*ALNKBR
            AKFB2     = EXP(ALNKFR)
            AKBB2     = EXP(ALNKBR)
            PROD1     = CONCEN(1)*CONCEN(3)
            PROD2     = CONCEN(2)*CONCEN(2)
            OMEGAF    = AKFB2*PROD1*AKFB2
            OMEGAB    = AKBB2*PROD2*AKBB2
            WREACT(1) = OMEGAF - OMEGAB
C
C           FIND NENSPEC FOR THIS REACTION
C
            DENFAC = 0.
            RMIN   = -10.

            IF (WREACT(1) .LT. 0.) THEN
C
C              NENSPEC IS OH
C
               IF (CONCEN(2) .GT. 1.E-6) THEN
                  ROM   = 2.*WREACT(1)/CONCEN(2)
                  IF (ROM .LT. RMIN) DENFAC = SONDPR/CONCEN(2)*2.*OMEGAB
               ENDIF
C
            ELSE
C
C              NENSPEC IS EITHER H2 OR O2
C
               IF (CONCEN(1) .GT. 1.E-6) THEN
                  ROM   = -WREACT(1)/CONCEN(1)
                  IF (ROM .LT. RMIN) THEN
                      RMIN = RMON
                      DENFAC = SONDPR/CONCEN(1)*OMEGAF
                  ENDIF
               ENDIF

               IF (CONCEN(3) .GT. 1.E-6) THEN
                  ROM   = -WREACT(1)/CONCEN(3)
                  IF (ROM .LT. RMIN) THEN
                      RMIN = RMON
                      DENFAC = SONDPR/CONCEN(3)*OMEGAF
                  ENDIF
               ENDIF
C
            ENDIF
C
C           ADJUST THE REACTION CONTRIBUTION FOR NENSPEC
C
            WREACT(1) = WREACT(1)/(1.+DENFAC)
C
C
C           REACTION # 2
C
            ALNKFR    = PREFCH(2) + EXPFCH(2)*ALOGT - ENEFCH(2)*RTEMP
            ALNKBR    = PREBCH(2) + EXPBCH(2)*ALOGT - ENEBCH(2)*RTEMP
            ALNKFR    = 0.5*ALNKFR
            ALNKBR    = 0.5*ALNKBR
```

```
            AKFB2     = EXP(ALNKFR)
            AKBB2     = EXP(ALNKBR)
            PROD1     = CONCEN(3)*CONCEN(2)*CONCEN(2)
            PROD2     = CONCEN(4)*CONCEN(4)
            OMEGAF    = AKFB2*PROD1*AKFB2
            OMEGAB    = AKBB2*PROD2*AKBB2
            WREACT(2) = OMEGAF - OMEGAB
C
C           FIND NENSPEC FOR THIS REACTION
C
            RMIN   = -10.
            DENFAC = 0.

            IF (WREACT(2) .LT. 0.) THEN
C
C              NENSPEC IS H2O
C
               IF (CONCEN(4) .GT. 1.E-6) THEN
                   ROM   = 2.*WREACT(2)/CONCEN(4)
                   IF (ROM .LT. RMIN) DENFAC = SONDPR/CONCEN(4)*2.*OMEGAB
               ENDIF
C
            ELSE
C
C              NENSPEC IS EITHER H2 OR OH
C
               IF (CONCEN(2) .GT. 1.E-6) THEN
                   ROM   = -2.*WREACT(2)/CONCEN(2)
                   IF (ROM .LT. RMIN) THEN
                        RMIN = RMON
                        DENFAC = SONDPR/CONCEN(2)*2.*OMEGAF
                   ENDIF
               ENDIF

               IF (CONCEN(3) .GT. 1.E-6) THEN
                   ROM   = -WREACT(2)/CONCEN(3)
                   IF (ROM .LT. RMIN) THEN
                        RMIN = RMON
                        DENFAC = SONDPR/CONCEN(3)*OMEGAF
                   ENDIF
               ENDIF
C
            ENDIF
C
C           ADJUST THE REACTION CONTRIBUTION FOR NENSPEC
C
            WREACT(2) = WREACT(2)/(1.+DENFAC)
C
C           COMPUTE THE SOURCE TERMS

            BIGWJA(5) =   -AMWTCH(1)*RECWDR* WREACT(1)
            BIGWJA(8) = 2.*AMWTCH(4)*RECWDR* WREACT(2)
            BIGWJA(6) = 2.*AMWTCH(2)*RECWDR*(WREACT(1)-WREACT(2))
            BIGWJA(7) =   -AMWTCH(3)*RECWDR*(WREACT(1)+WREACT(2))

            RETURN
          ENDIF
```

```
C
C       COMPUTE THE CONTRIBUTION WREACT TO THE SOURCE TERMS FROM ALL
C       THE REACTIONS

        DO 50 IR = 1, NREACH
           ALNKFR = PREFCH(IR) + EXPFCH(IR)*ALOGT - ENEFCH(IR)*RTEMP
           ALNKBR = PREBCH(IR) + EXPBCH(IR)*ALOGT - ENEBCH(IR)*RTEMP
           ALNKFR = 0.5*ALNKFR
           ALNKBR = 0.5*ALNKBR
           AKFB2  = EXP(ALNKFR)
           AKBB2  = EXP(ALNKBR)
           PROD1  = 1.DO
           PROD2  = 1.DO
           NSRK   = NSRKCH(IR)
           DO 30 IS = 1, NSRK
             ISP   = ITABCH(IS ,IR)
             IP1   = IALOCH(ISP,IR)
             IP2   = IBTOCH(ISP,IR)
             IF (IP1 .NE. 0) PROD1 = PROD1*CONCEN(ISP)**IP1
             IF (IP2 .NE. 0) PROD2 = PROD2*CONCEN(ISP)**IP2
30         CONTINUE
           OMEGAF     = AKFB2*PROD1*AKFB2
           OMEGAB     = AKBB2*PROD2*AKBB2
           WREACT(IR) = OMEGAF - OMEGAB
C
C          FIND NENSPEC FOR THIS REACTION
C
           RMIN   = -10.
           DENFAC = 0.

           DO 40 IS = 1, NSRK
              ISP   = ITABCH(IS ,IR)
              IF (CONCEN(ISP) .GT. 1.E-6) THEN
                 ROM   = BMIACH(IS,IR)*WREACT(IR)/CONCEN(ISP)
                 IF (ROM .LT. RMIN) THEN
                    IP1  = IALPCH(ISP,IR)
                    IP2  = IBETCH(ISP,IR)
                    RMIN = RMON
                    DENFAC = SONDPR/CONCEN(ISP)*(IP1*OMEGAF+IP2*OMEGAB)
                 ENDIF
              ENDIF
40         CONTINUE
C
C          ADJUST THE REACTION CONTRIBUTION FOR NENSPEC
C
           WREACT(IR) = WREACT(IR)/(1.+DENFAC)
C
50      CONTINUE

C       COMPUTE THE SOURCE TERMS

        DO 70 IS = 1, NEQSCH
           JS     = NEQBAS + IS
           SUMWT  = 0.
           DO 60 IR = 1, NREACH
             SUMWT  = SUMWT + BMIACH(IS,IR)*WREACT(IR)
60         CONTINUE
```

```
            BIGWJA(JS) = AMWTCH(IS)*SUMWT
            IF (KROGER .EQ. 2) BIGWJA(JS) = BIGWJA(JS)*RHOD
            BIGWJA(JS) = BIGWJA(JS)*RECWDR
70      CONTINUE

        RETURN
        END
```

E2SOUR

```
        SUBROUTINE E2SOUR

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] CHCOMN.INC/LIST'
        INCLUDE '[.INC] E2COMN.INC/LIST'
        INCLUDE '[.INC] FLCOMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        INCLUDE '[.INC] JACOMN.INC/LIST'
        INCLUDE '[.INC] PRCOMN.INC/LIST'
        DIMENSION WREACT(MREACH)
        DOUBLE PRECISION PROD1, PROD2, CONCEN(MSPECH)

C***********************************************************************

C       THIS FUNCTION COMPUTES THE SOURCE TERMS SO THAT THE JACOBIAN
C       TERMS COULD BE COMPUTED.  IT ALSO COMPUTES SOME FLUX TERMS, ONLY
C       TWO FLUX TERMS, F2, F4 AND G3, G4 ARE NEEDED FOR NUMERICAL
C       COMPUTATION OF FLUX JACOBIANS, WHEREAS ALL THE SOURCE TERMS ARE
C       NEEDED FOR SOURCE JACOBIANS.  THE TEMPORALLY VARYING VARIABLES ARE
C       STORED IN THE JA COMMON VARIABLES, I.E.,
C                    DPENJA(J) = DPENG1(J,INODE)
C       FOR THE GIVEN NODE INODE.
C       THE DIFFERENCE BETWEEN THIS ROUTINE AND E2FLUX IS THAT IN THAT
C       ROUTINE WE DO NOT CONSIDER VARIATIONS W.R.T. STATE VARIABLES.

C***********************************************************************

        RHO    = DPENJA(1)
        UCOMP  = DPENJA(2)/DPENJA(1)
        VCOMP  = DPENJA(3)/DPENJA(1)
        BEPS   = DPENJA(4)
        BEU    = BEPS/RHO
        VELO2U = UCOMP*UCOMP + VCOMP*VCOMP
C
C       COMPUTE THE DIMENSIONAL QUANTITIES
C
        BE     = FMREFL*BEU
        VELO2  = FMREFL*VELO2U

C       COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

        SUMY = 0.
```

```
      DO 5 IS = 1, NEQSCH
         JS          = NEQBAS + IS
         YSPEPR(IS) = DPENJA(JS)/DPENJA(1)
         SUMY        = SUMY + YSPEPR(IS)
5     CONTINUE

      YNEXT             = 1. - SUMY - YNRTCH
      IF (YNEXT .LT. 0.) YNEXT = 0.
      YSPEPR(NEQSCH+1) = YNEXT

      SYSHFS = 0.
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
C     COMPUTE THE TEMPERATURE IN DEGREE K AND ALSO
C     COMPUTE THE CONCENTRATIONS OF ALL THE SPECIES IN KMOL/(M**3)

      DO 10 IS = 1, NSPECH
         SYSHFS     = SYSHFS + YSPEPR(IS)*FMHTCH(IS)/AMWTCH(IS)
         SYSCPS     = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
         SYSBMS     = SYSBMS + YSPEPR(IS)/AMWTCH(IS)
         BIGAM      = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
         CONCEN(IS) = RHO*RHORFL*YSPEPR(IS)/AMWTCH(IS)
10    CONTINUE
C
C     COMPUTE TEMPERATURE IN DEGREE K AND SOME RELATED QUANTITIES

      BIGBM   = SYSCPS - UGASFL*SYSBMS
      BIGCM   = BE - 0.5*VEL02 - SYSHFS + TREFCH*SYSCPS
     1                + 0.5*TREFCH*TREFCH*BIGAM
      IF (BIGAM .LT. 1.E-10) THEN
         TEMP   = BIGCM/BIGBM
      ELSE
         DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
         TEMP   = ( SQRT(DISCRI)-BIGBM )/BIGAM
      ENDIF
      ALOGT  = LOG(TEMP)
      RTEMP  = 1./TEMP
C
C     NORMALIZE THE TEMPERATURE
C
      TEMPU = TEMP/TREFFL
C
C     COMPUTE THE DIMENSIONLESS PRESSURE
C
      PRESS = RHO*TEMPU*AMWTFL*SYSBMS

C     COMPUTE THE FLUX VARIABLES (FIRST FOUR)

      BGF2JA = DPENJA(2)*UCOMP + PRESS
      BGG3JA = DPENJA(3)*VCOMP + PRESS

      BGF4JA = (BEPS+PRESS)*UCOMP
      BGG4JA = (BEPS+PRESS)*VCOMP
```

```
C       BY-PASS THE REACTION CALCULATIONS IF TEMPERATURE IS LESS THAN
C       TRIGGER TEMPERATURE

        FROZEN = 1.
        IF (TEMP .LT. TRIGCH) THEN
          FROZEN = 0.
          GO TO 45
        ENDIF
C
C       CORRECT THE RATE COEFFICIENTS FOR ROGERS AND CHINITZ MODEL
C
        IF (KROGER .EQ. 1) THEN
           PHI       =  YSPEPR(3)*34.048/(1.-YSPEPR(3))
           IF (PHI .LT. 0.1 ) PHI = 0.1
           IF (PHI .GT. 2.0 ) PHI = 2.0
           RPHI      =  1./PHI
           TENLOG    =  LOG(10.)
           A1PHI     =  8.917*PHI + 31.433*RPHI - 28.95
           A2PHI     = -0.833*PHI +  1.333*RPHI +  2.00
           PREFCH(1) =  LOG(A1PHI) + 44.*TENLOG
           PREFCH(2) =  LOG(A2PHI) + 58.*TENLOG
           PREBCH(1) =  PREFCH(1) - PREECH(1)
           PREBCH(2) =  PREFCH(2) - PREECH(2)
        ENDIF
C
C       COMPUTE THE CONTRIBUTION WREACT TO THE SOURCE TERMS FROM ALL
C       THE REACTIONS

        DO 40 IR = 1, NREACH
          ALNKFR = PREFCH(IR) + EXPFCH(IR)*ALOGT - ENEFCH(IR)*RTEMP
          ALNKBR = PREBCH(IR) + EXPBCH(IR)*ALOGT - ENEBCH(IR)*RTEMP
          ALNKFR = 0.5*ALNKFR
          ALNKBR = 0.5*ALNKBR
          AKFB2  = EXP(ALNKFR)
          AKBB2  = EXP(ALNKBR)
          PROD1  = 1.D0
          PROD2  = 1.D0
          NSRK   = NSRKCH(IR)
          DO 30 IS = 1, NSRK
            ISP   = ITABCH(IS ,IR)
            IP1   = IALOCH(ISP,IR)
            IP2   = IBTOCH(ISP,IR)
            IF (IP1 .NE. 0) PROD1 = PROD1*CONCEN(ISP)**IP1
            IF (IP2 .NE. 0) PROD2 = PROD2*CONCEN(ISP)**IP2
30        CONTINUE
          OMEGAF     = AKFB2*PROD1*AKFB2
          OMEGAB     = AKBB2*PROD2*AKBB2
          WREACT(IR) = OMEGAF - OMEGAB
40      CONTINUE

C       COMPUTE THE SOURCE TERMS

45      DO 60 IS = 1, NEQSCH
          JS     = NEQBAS + IS
          SUMWT  = 0.
          DO 50 IR = 1, NREACH
            SUMWT  = SUMWT + BMA_CH(IS,IR)*WREACT(IR)
```

```
50        CONTINUE
          BIGWJA(JS) = AMWTCH(IS)*SUMWT*FROZEN
          IF (KROGER .EQ. 2) BIGWJA(JS) = BIGWJA(JS)*RHO*RHORFL
          BIGWJA(JS) = BIGWJA(JS)/WDREFL
60      CONTINUE
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGE2 .NE. 8 .AND. IDBGE2 .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) BGF2JA,BGF4JA,BGG3JA,BGG4JA

        DO 70 IS = 1, NEQNFL
          WRITE(JDEBUG,1400) IS, DPENJA(IS), BIGWJA(IS)
70      CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM E2SOUR' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(  5X, 'BGF2JA=', G14.5, 5X, 'BGF4JA=', G14.5/
     1           5X, 'BGG3JA=', G14.5, 5X, 'BGG4JA=', G14.5/
     2           5X, 'IS', 5X, 'DPENJA',10X,'BIGWJA')
1400    FORMAT(5X,I5,2G14.5)

        RETURN
        END
```

E2TIMU

```
        SUBROUTINE E2TIMO
C                  E2TIMU

        PARAMETER (GAMMAX=1.66, ZBASLG=0.69314718, TENLOG=2.302585093)
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'CHCOMN.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'FLCOMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'HEXCOD.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'JACOMN.INC'
        INCLUDE 'M2COMN.INC'
        INCLUDE 'PRCOMN.INC'
```

```
      INCLUDE 'TICOMN.INC'

      DIMENSION NCEL(MMAXTI), ICELTT(MMAXTI,MCELG2), CELLBN(MMAXTI)
      DIMENSION WREACT(MREACH)
      DOUBLE PRECISION PROD1, PROD2, CONCEN(MSPECH)

C*********************************************************************

C     THIS SUBROUTINE STEPS THROUGH EACH CELL ON THIS LEVEL AND
C     COMPUTES THE CELL TIME STEP AS A FIRST STEP.  THE CELL TIME-
C     STEPS ARE REASSIGNED AS MULTIPLES OF INTEGRAL POWERS OF 2
C     TIMES THE GLOBAL MIMIMUM TIME-STEP IF ATLEAST A FACTOR OF
C     2 EXISTS BETWEEN GLOBAL MIMIMUM AND GLOBAL MAXIMUM VALUES.

C*********************************************************************

C     INITIALIZE QUANTITIES

      DTMNTI = 1000.
      DTMAX  = 0.
      MAXCHR = 4
      kadphr = kadpti
      if (kadphr .eq. 99) kadphr = 0
C     ZBASLG = LOG(2.)

C     STEP THROUGH ALL THE CEWIC CELLS AND FIND CELL TIMESTEPS
C
C     -------------
C     CFL CONDITION
C     -------------
C
CVD$  NOLSTVAL
CVD$  NODEPCHK
      DO 10 JCELL = 1, NCELA2

C        ----------------------
C        CELL/NODE DETERMINATION
C        ----------------------

C        FIND THE ACTUAL CELL NUMBER
         ICELL = ICELA2(JCELL)

C        SET UP NODE POINTERS FOR THIS CELL

         KSW = ICELG2(2,ICELL)
         KSE = ICELG2(4,ICELL)
         KNE = ICELG2(6,ICELL)
         KNW = ICELG2(8,ICELL)
C
C        -----------------
C        CELL CENTER VALUES
C        -----------------
C
C        DETERMINE THE DEPENDENT VARIABLES (DENSITY, PRESSURE, AND
C        VELOCITY COMPONENTS) AT THE CENTER OF THE CELL

C        AVERAGE VALUES AT THE CENTER
```

```
      DPENJA(1) = 0.25*( DPENG2(1,KSW) + DPENG2(1,KSE)
     1             + DPENG2(1,KNE) + DPENG2(1,KNW) )
      DPENJA(2) = 0.25*( DPENG2(2,KSW) + DPENG2(2,KSE)
     1             + DPENG2(2,KNE) + DPENG2(2,KNW) )
      DPENJA(3) = 0.25*( DPENG2(3,KSW) + DPENG2(3,KSE)
     1             + DPENG2(3,KNE) + DPENG2(3,KNW) )

C     GET THE PRESSURE FOR THE CELL

      PRESPR = 0.25*( PRESG2(KSW) + PRESG2(KSE) + PRESG2(KNE)
     1                            + PRESG2(KNW)             )

C     COMPUTE THE VELOCITY COMPONENTS AND SPEED OF SOUND

      UCOMPC = DPENJA(2)/DPENJA(1)
      VCOMPC = DPENJA(3)/DPENJA(1)
      SOUND  = ABS(GAMMAX*PRESPR/DPENJA(1))
      SOUND  = SQRT(SOUND)

C     COMPUTE AVERAGE DISTANCES

      DISTEW = SQRT(DXEWM2(ICELL)*DXEWM2(ICELL) +
     1              DYEWM2(ICELL)*DYEWM2(ICELL) )
      DISTNS = SQRT(DXNSM2(ICELL)*DXNSM2(ICELL) +
     1              DYNSM2(ICELL)*DYNSM2(ICELL) )

C     COMPUTE THE CFL CONDITION IN THE TWO DIRECTIONS

      DTEW   = ABS(UCOMPC*DYEWM2(ICELL) - VCOMPC*DXEWM2(ICELL))
     1                                  + SOUND*DISTEW
      DTNS   = ABS(UCOMPC*DYNSM2(ICELL) - VCOMPC*DXNSM2(ICELL))
     1                                  + SOUND*DISTNS
      EIGEN  = MAX(DTEW,DTNS)

C     FOR COARSER CELLS INCREASE THE CFL NUMBER BY A FACTOR OF
C     TWO FOR EACH SPATIAL LEVEL CAORSENING
      KX     = KAUXG2(ICELL)
      K5LEVG = IAND(KX,KU000F)
      LEVELG = ISHFT(K5LEVG,-16)
      ILEVEL = NLVLG2 - LEVELG
      IFACTR = 2**ILEVEL
      CFLTT  = CFLNTI*IFACTR
      CFLTT  = MIN (CFLXTI,CFLTT)
C
      CELLTI(ICELL) = CFLTT*FCTRTI/(EIGEN*RVOLM2(ICELL))
C
10    CONTINUE
C
C     ------------------
C     LOCAL CFL CONDITION
C     ------------------
C
C     THIS IS AN ATTEMPT TO HANDLE STEADY-STATE PROBLEMS
C
      IF (KADPTI .EQ. 99) THEN
```

```
            NGIVTI = 0
            NMAXTI = 0
            -
            DO 15 JCELL    = 1, NCELA2
               ICELL         = ICELA2(JCELL)
               ICELTI(JCELL) = ICELL
15          CONTINUE

            ILVLTI(1,0) = 1
            ILVLTI(2,0) = NCELA2
            RETURN

         ENDIF
C
C        ------------------
C        TEMPORAL RESOLUTION
C        ------------------
C
C        SKIP THE TEMPORAL RESOLUTION IF KADPTI EQUALS ZERO OR USE THE
C        APPROPRIATE FLUXES FOR CONTINUITY, MOMENTA OR ENERGY EQUATIONS
C
         GOTO (190, 20, 40, 60, 80), KADPTI+1
C
C        IF (KADPTI .GT. NEQNFL) GOTO 190
C        USE SPECIES EQUATION FOR TEMPORAL RESOLUTION
         GOTO 100
C
C        RESOLUTION BASED UPON CONTINUITY EQUATION

CVD$     NOLSTVAL
CVD$     NODEPCHK
20       DO 30 JCELL = 1, NCELA2

C           FIND THE ACTUAL CELL NUMBER
            ICELL = ICELA2(JCELL)

C           SET UP NODE POINTERS FOR THIS CELL
            KSW = ICELG2(2,ICELL)
            KSE = ICELG2(4,ICELL)
            KNE = ICELG2(6,ICELL)
            KNW = ICELG2(8,ICELL)

C           GEOMETRY OF ALL CELL CORNERS
            XSW = GEOMG2(1,KSW)
            YSW = GEOMG2(2,KSW)
            XSE = GEOMG2(1,KSE)
            YSE = GEOMG2(2,KSE)
            XNE = GEOMG2(1,KNE)
            YNE = GEOMG2(2,KNE)
            XNW = GEOMG2(1,KNW)
            YNW = GEOMG2(2,KNW)
C
C           DETERMINE THE DEPENDENT VARIABLES AT THE CENTER OF THE CELL
            DPENJA(KADPTI) = 0.25*(  DPENG2(KADPTI,KSW)
     1                             + DPENG2(KADPTI,KSE)
     2                             + DPENG2(KADPTI,KNE)
     3                             + DPENG2(KADPTI,KNW) )
```

```
C          NOW COMPUTE THE FLUX TERMS AT THE FOUR CORNER NODES
C          CORRESPONDING TO THE CONTINUITY EQUATION

           BIGFSW  = DPENG2(2,KSW)
           BIGFSE  = DPENG2(2,KSE)
           BIGFNE  = DPENG2(2,KNE)
           BIGFNW  = DPENG2(2,KNW)

           BIGGSW  = DPENG2(3,KSW)
           BIGGSE  = DPENG2(3,KSE)
           BIGGNE  = DPENG2(3,KNE)
           BIGGNW  = DPENG2(3,KNW)

C          COMPUTE THE FIRST ORDER CHANGE (/CELLTI)
           DENO  = (BIGFSW-BIGFNE)*(YNW-YSE) +
     1             (BIGFNW-BIGFSE)*(YNE-YSW) +
     2             (BIGGSW-BIGGNE)*(XSE-XNW) +
     3             (BIGGNW-BIGGSE)*(XSW-XNE)

           DENO  = 0.5*DENO*RVOLM2(ICELL)
           DTR   = 1000.
           IF (DENO .NE. 0.)
     1          DTR = (EPSOTI+EPS1TI*DPENJA(KADPTI))/ABS(DENO)
           CELLTI(ICELL) = MIN (CELLTI(ICELL), DTR)

30       CONTINUE

         GOTO 190

C        RESOLUTION BASED UPON X-MOMENTUM EQUATION

CVD$     NOLSTVAL
CVD$     NODEPCHK
40       DO 50 JCELL = 1, NCELA2

C           FIND THE ACTUAL CELL NUMBER
            ICELL = ICELA2(JCELL)

C           SET UP NODE POINTERS FOR THIS CELL
            KSW = ICELG2(2,ICELL)
            KSE = ICELG2(4,ICELL)
            KNE = ICELG2(6,ICELL)
            KNW = ICELG2(8,ICELL)

C           GEOMETRY OF ALL CELL CORNERS
            XSW = GEOMG2(1,KSW)
            YSW = GEOMG2(2,KSW)
            XSE = GEOMG2(1,KSE)
            YSE = GEOMG2(2,KSE)
            XNE = GEOMG2(1,KNE)
            YNE = GEOMG2(2,KNE)
            XNW = GEOMG2(1,KNW)
            YNW = GEOMG2(2,KNW)
C
C           DETERMINE THE DEPENDENT VARIABLES AT THE CENTER OF THE CELL
            DPENJA(KADPTI) = 0.25*(  DPENG2(KADPTI,KSW)
```

```
     1                          + DPENG2(KADPTI,KSE)
     2                          + DPENG2(KADPTI,KNE)
     3                          + DPENG2(KADPTI,KNW) )

C          NOW COMPUTE THE FLUX TERMS AT THE FOUR CORNER NODES
C          CORRESPONDING TO THE X-MOMENTUM EQUATION

           BIGFSW  = DPENG2(2,KSW)*DPENG2(2,KSW)/DPENG2(1,KSW) +
     1               PRESG2(KSW)
           BIGFSE  = DPENG2(2,KSE)*DPENG2(2,KSE)/DPENG2(1,KSE) +
     1               PRESG2(KSE)
           BIGFNE  = DPENG2(2,KNE)*DPENG2(2,KNE)/DPENG2(1,KNE) +
     1               PRESG2(KNE)
           BIGFNW  = DPENG2(2,KNW)*DPENG2(2,KNW)/DPENG2(1,KNW) +
     1               PRESG2(KNW)

           BIGGSW  = DPENG2(2,KSW)*DPENG2(3,KSW)/DPENG2(1,KSW)
           BIGGSE  = DPENG2(2,KSE)*DPENG2(3,KSE)/DPENG2(1,KSE)
           BIGGNE  = DPENG2(2,KNE)*DPENG2(3,KNE)/DPENG2(1,KNE)
           BIGGNW  = DPENG2(2,KNW)*DPENG2(3,KNW)/DPENG2(1,KNW)

C          COMPUTE THE FIRST ORDER CHANGE (/CELLTI)
           DENO  = (BIGFSW-BIGFNE)*(YNW-YSE) +
     1             (BIGFNW-BIGFSE)*(YNE-YSW) +
     2             (BIGGSW-BIGGNE)*(XSE-XNW) +
     3             (BIGGNW-BIGGSE)*(XSW-XNE)

           DENO  = 0.5*DENO*RVOLM2(ICELL)
           DTR   = 1000.
           IF (DENO .NE. 0.)
     1               DTR = (EPSOTI+EPS1TI*DPENJA(KADPTI))/ABS(DENO)
           CELLTI(ICELL) = MIN (CELLTI(ICELL), DTR)

50      CONTINUE

        GOTO 190

C       RESOLUTION BASED UPON Y-MOMENTUM EQUATION

CVD$    NOLSTVAL
CVD$    NODEPCHK
60      DO 70 JCELL = 1, NCELA2

C          FIND THE ACTUAL CELL NUMBER
           ICELL = ICELA2(JCELL)

C          SET UP NODE POINTERS FOR THIS CELL
           KSW = ICELG2(2,ICELL)
           KSE = ICELG2(4,ICELL)
           KNE = ICELG2(6,ICELL)
           KNW = ICELG2(8,ICELL)

C          GEOMETRY OF ALL CELL CORNERS
           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)
           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)
```

```
            XNE = GEOMG2(1,KNE)
            YNE = GEOMG2(2,KNE)
            XNW = GEOMG2(1,KNW)
            YNW = GEOMG2(2,KNW)
C
C           DETERMINE THE DEPENDENT VARIABLES AT THE CENTER OF THE CELL
            DPENJA(KADPTI) = 0.25*(  DPENG2(KADPTI,KSW)
     1                             + DPENG2(KADPTI,KSE)
     2                             + DPENG2(KADPTI,KNE)
     3                             + DPENG2(KADPTI,KNW) )

C           NOW COMPUTE THE FLUX TERMS AT THE FOUR CORNER NODES
C           CORRESPONDING TO THE X-MOMENTUM EQUATION

            BIGFSW  = DPENG2(2,KSW)*DPENG2(3,KSW)/DPENG2(1,KSW)
            BIGFSE  = DPENG2(2,KSE)*DPENG2(3,KSE)/DPENG2(1,KSE)
            BIGFNE  = DPENG2(2,KNE)*DPENG2(3,KNE)/DPENG2(1,KNE)
            BIGFNW  = DPENG2(2,KNW)*DPENG2(3,KNW)/DPENG2(1,KNW)

            BIGGSW  = DPENG2(3,KSW)*DPENG2(3,KSW)/DPENG2(1,KSW) +
     1                PRESG2(KSW)
            BIGGSE  = DPENG2(3,KSE)*DPENG2(3,KSE)/DPENG2(1,KSE) +
     1                PRESG2(KSE)
            BIGGNE  = DPENG2(3,KNE)*DPENG2(3,KNE)/DPENG2(1,KNE) +
     1                PRESG2(KNE)
            BIGGNW  = DPENG2(3,KNW)*DPENG2(3,KNW)/DPENG2(1,KNW) +
     1                PRESG2(KNW)

C           COMPUTE THE FIRST ORDER CHANGE (/CELLTI)
            DENO  = (BIGFSW-BIGFNE)*(YNW-YSE) +
     1              (BIGFNW-BIGFSE)*(YNE-YSW) +
     2              (BIGGSW-BIGGNE)*(XSE-XNW) +
     3              (BIGGNW-BIGGSE)*(XSW-XNE)

            DENO  = 0.5*DENO*RVOLM2(ICELL)
            DTR   = 1000.
            IF (DENO .NE. 0.)
     1                DTR = (EPSOTI+EPS1TI*DPENJA(KADPTI))/ABS(DENO)
            CELLTI(ICELL) = MIN (CELLTI(ICELL), DTR)

70      CONTINUE

        GOTO 190

C       RESOLUTION BASED UPON ENERGY EQUATION

CVD$    NOLSTVAL
CVD$    NODEPCHK
80      DO 90 JCELL = 1, NCELA2

C           FIND THE ACTUAL CELL NUMBER
            ICELL = ICELA2(JCELL)

C           SET UP NODE POINTERS FOR THIS CELL
            KSW = ICELG2(2,ICELL)
            KSE = ICELG2(4,ICELL)
            KNE = ICELG2(6,ICELL)
```

```
            KNW = ICELG2(8,ICELL)

C           GEOMETRY OF ALL CELL CORNERS
            XSW = GEOMG2(1,KSW)
            YSW = GEOMG2(2,KSW)
            XSE = GEOMG2(1,KSE)
            YSE = GEOMG2(2,KSE)
            XNE = GEOMG2(1,KNE)
            YNE = GEOMG2(2,KNE)
            XNW = GEOMG2(1,KNW)
            YNW = GEOMG2(2,KNW)
C
C           DETERMINE THE DEPENDENT VARIABLES AT THE CENTER OF THE CELL
            DPENJA(KADPTI) = 0.25*(  DPENG2(KADPTI,KSW)
     1                             + DPENG2(KADPTI,KSE)
     2                             + DPENG2(KADPTI,KNE)
     3                             + DPENG2(KADPTI,KNW) )

C           NOW COMPUTE THE FLUX TERMS AT THE FOUR CORNER NODES
C           CORRESPONDING TO THE X-MOMENTUM EQUATION

            BIGFSW  = ( DPENG2(4,KSW) + PRESG2(KSW) )*
     1                  DPENG2(2,KSW)/DPENG2(1,KSW)
            BIGFSE  = ( DPENG2(4,KSE) + PRESG2(KSE) )*
     1                  DPENG2(2,KSE)/DPENG2(1,KSE)
            BIGFNE  = ( DPENG2(4,KNE) + PRESG2(KNE) )*
     1                  DPENG2(2,KNE)/DPENG2(1,KNE)
            BIGFNW  = ( DPENG2(4,KNW) + PRESG2(KNW) )*
     1                  DPENG2(2,KNW)/DPENG2(1,KNW)

            BIGGSW  = ( DPENG2(4,KSW) + PRESG2(KSW) )*
     1                  DPENG2(3,KSW)/DPENG2(1,KSW)
            BIGGSE  = ( DPENG2(4,KSE) + PRESG2(KSE) )*
     1                  DPENG2(3,KSE)/DPENG2(1,KSE)
            BIGGNE  = ( DPENG2(4,KNE) + PRESG2(KNE) )*
     1                  DPENG2(3,KNE)/DPENG2(1,KNE)
            BIGGNW  = ( DPENG2(4,KNW) + PRESG2(KNW) )*
     1                  DPENG2(3,KNW)/DPENG2(1,KNW)

C           COMPUTE THE FIRST ORDER CHANGE (/CELLTI)
            DENO  = (BIGFSW-BIGFNE)*(YNW-YSE) +
     1              (BIGFNW-BIGFSE)*(YNE-YSW) +
     2              (BIGGSW-BIGGNE)*(XSE-XNW) +
     3              (BIGGNW-BIGGSE)*(XSW-XNE)

            DENO  = 0.5*DENO*RVOLM2(ICELL)
            DTR   = 1000.
            IF (DENO .NE. 0.)
     1             DTR = (EPSOTI+EPS1TI*DPENJA(KADPTI))/ABS(DENO)
            CELLTI(ICELL) = MIN (CELLTI(ICELL), DTR)

90      CONTINUE

        GOTO 190
C
C       USE SPECIES EQUATION FOR TEMPORAL RESOLUTION
```

```
100     IF (KDIFTI .NE. 0) GOTO 181

        DO 180 JCELL = 1, NCELA2

C          FIND THE ACTUAL CELL NUMBER
           ICELL = ICELA2(JCELL)

C          SET UP NODE POINTERS FOR THIS CELL
           KSW = ICELG2(2,ICELL)
           KSE = ICELG2(4,ICELL)
           KNE = ICELG2(6,ICELL)
           KNW = ICELG2(8,ICELL)

C          GEOMETRY OF ALL CELL CORNERS
           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)
           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)
           XNE = GEOMG2(1,KNE)
           YNE = GEOMG2(2,KNE)
           XNW = GEOMG2(1,KNW)
           YNW = GEOMG2(2,KNW)
C
C          DETERMINE ALL THE DEPENDENT VARIABLES AT THE CENTER
C          OF THE CELL
C
           DO 110 IQ = 1, NEQNFL
               DPENJA(IQ) = 0.25*( DPENG2(IQ,KSW) + DPENG2(IQ,KSE)
     1                           + DPENG2(IQ,KNE) + DPENG2(IQ,KNW) )
110        CONTINUE

C          GET THE TEMPERATURE FOR THE CELL
           TEMPPR = 0.25*( TEMPG2(KSW) + TEMPG2(KSE) +
     1                     TEMPG2(KNE) + TEMPG2(KNW) )

C          COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
           SUMY = 0.

           DO 120 IS = 1, NEQSCH
              JS         = NEQBAS + IS
              YSPEPR(IS) = DPENJA(JS)/DPENJA(1)
              SUMY       = SUMY + YSPEPR(IS)
120        CONTINUE

           YNEXT            = 1. - SUMY - YNRTCH
           IF (YNEXT .LT. 0.) YNEXT = 0.
           YSPEPR(NEQSCH+1) = YNEXT

C          COMPUTE THE CONCENTRATIONS OF ALL THE SPECIES IN KMOL/(M**3)

           DO 130 IS = 1, NSPECH
              CONCEN(IS) = DPENJA(1)*RHORFL*YSPEPR(IS)*RAMWCH(IS)
130        CONTINUE
C
C          COMPUTE TEMPERATURE IN DEGREE K AND SOME RELATED QUANTITIES

           TEMP   = TEMPPR*TREFFL
```

```
      ALOGT  = LOG(TEMP)
      RTEMP  = 1./TEMP
C
C     BY-PASS THE REACTION CALCULATIONS IF TEMPERATURE IS LESS
C     THAN TRIGGER TEMPERATURE

      IF (TEMP .LT. TRIGCH) THEN
          BIGWCE = 0.
          GOTO 170
      ENDIF
C
C     CORRECT THE RATE COEFFICIENTS FOR ROGERS AND CHINITZ MODEL
C
      IF (KROGER .EQ. 1) THEN
         PHI       =  YSPEPR(3)*34.048/(1.-YSPEPR(3))
         IF (PHI .LT. 0.1 ) PHI = 0.1
         IF (PHI .GT. 2.0 ) PHI = 2.0
         RPHI      =  1./PHI
C        TENLOG    =  LOG(10.)
         A1PHI     =  8.917*PHI + 31.433*RPHI - 28.95
         A2PHI     = -0.833*PHI +  1.333*RPHI +  2.00
         PREFCH(1) =  LOG(A1PHI) + 44.*TENLOG
         PREFCH(2) =  LOG(A2PHI) + 58.*TENLOG
         PREBCH(1) =  PREFCH(1) - PREECH(1)
         PREBCH(2) =  PREFCH(2) - PREECH(2)
      ENDIF
C
C     COMPUTE THE CONTRIBUTION WREACT TO THE SOURCE TERMS
C     FROM ALL THE REACTIONS

      DO 150 IR = 1, NREACH
         ALNKFR = PREFCH(IR) +EXPFCH(IR)*ALOGT -ENEFCH(IR)*RTEMP
         ALNKBR = PREBCH(IR) +EXPBCH(IR)*ALOGT -ENEBCH(IR)*RTEMP
         ALNKFR = 0.5*ALNKFR
         ALNKBR = 0.5*ALNKBR
         AKFB2  = EXP(ALNKFR)
         AKBB2  = EXP(ALNKBR)
         PROD1  = 1.DO
         PROD2  = 1.DO
         NSRK   = NSRKCH(IR)
         DO 140 IS = 1, NSRK
           ISP   = ITABCH(IS ,IR)
           IP1   = IALOCH(ISP,IR)
           IP2   = IBTOCH(ISP,IR)
           IF (IP1 .NE. 0) PROD1 = PROD1*CONCEN(ISP)**IP1
           IF (IP2 .NE. 0) PROD2 = PROD2*CONCEN(ISP)**IP2
140      CONTINUE
         OMEGAF     = AKFB2*PROD1*AKFB2
         OMEGAB     = AKBB2*PROD2*AKBB2
         WREACT(IR) = OMEGAF - OMEGAB
150   CONTINUE

C     COMPUTE THE SOURCE TERMS

      SUMWT  = 0.
      DO 160 IR = 1, NREACH
         SUMWT  = SUMWT + BMIACH(IS,IR)*WREACT(IR)
```

```
160         CONTINUE
            BIGWCE = AMWTCH(IS)*SUMWT/WDREFL
            IF (KROGER .EQ. 2) BIGWCE = BIGWCE*DPENJA(1)*RHORFL

C           NOW COMPUTE THE FLUX TERMS AT THE FOUR CORNER NODES
C           CORRESPONDING TO THE X-MOMENTUM EQUATION

170         BIGFSW  = DPENG2(KADPTI,KSW)*DPENG2(2,KSW)/DPENG2(1,KSW)
            BIGFSE  = DPENG2(KADPTI,KSE)*DPENG2(2,KSE)/DPENG2(1,KSE)
            BIGFNE  = DPENG2(KADPTI,KNE)*DPENG2(2,KNE)/DPENG2(1,KNE)
            BIGFNW  = DPENG2(KADPTI,KNW)*DPENG2(2,KNW)/DPENG2(1,KNW)

            BIGGSW  = DPENG2(KADPTI,KSW)*DPENG2(3,KSW)/DPENG2(1,KSW)
            BIGGSE  = DPENG2(KADPTI,KSE)*DPENG2(3,KSE)/DPENG2(1,KSE)
            BIGGNE  = DPENG2(KADPTI,KNE)*DPENG2(3,KNE)/DPENG2(1,KNE)
            BIGGNW  = DPENG2(KADPTI,KNW)*DPENG2(3,KNW)/DPENG2(1,KNW)

C           COMPUTE THE FIRST ORDER CHANGE (/CELLTI)
            DENO  = (BIGFSW-BIGFNE)*(YNW-YSE) +
     1              (BIGFNW-BIGFSE)*(YNE-YSW) +
     2              (BIGGSW-BIGGNE)*(XSE-XNW) +
     3              (BIGGNW-BIGGSE)*(XSW-XNE)

            DENO  = BIGWCE + 0.5*DENO*RVOLM2(ICELL)
            DTR   = 1000.
            IF (DENO .NE. 0.)
     1             DTR = (EPSOTI+EPS1TI*DPENJA(KADPTI))/ABS(DENO)
            CELLTI(ICELL) = MIN (CELLTI(ICELL), DTR)

180      CONTINUE

         GOTO 190

181      DO 185 JCELL = 1, NCELA2

C           FIND THE ACTUAL CELL NUMBER
            ICELL = ICELA2(JCELL)

C           SET UP NODE POINTERS FOR THIS CELL
            KSW = ICELG2(2,ICELL)
            KSE = ICELG2(4,ICELL)
            KNE = ICELG2(6,ICELL)
            KNW = ICELG2(8,ICELL)
C
C           DETERMINE THE MASS FRACTIONS AT THE CORNERS
C
            YSPKSW = DPENG2(KADPTI,KSW)/DPENG2(1,KSW)
            YSPKSE = DPENG2(KADPTI,KSE)/DPENG2(1,KSE)
            YSPKNE = DPENG2(KADPTI,KNE)/DPENG2(1,KNE)
            YSPKNW = DPENG2(KADPTI,KNW)/DPENG2(1,KNW)
C
C           DETERMINE THE MAX/MIN MASS FRACTION VARIATIONS
C
            YSPMAX = MAX (YSPKSW, YSPKSE, YSPKNE, YSPKNW)
            YSPMIN = MIN (YSPKSW, YSPKSE, YSPKNE, YSPKNW)
            YSPMIN = YSPMAX - YSPMIN
            IF (YSPMIN .LE. 1.E-5) GOTO 185
```

```
C
C          THE GREATER THE DIFFERENCE OF MASS FRACTIONS AMONG NEIGHBOURING
C          CELLS THE GREATER WILL BE THE FACTOR BY WHICH THE CELL TIMESTEP
C          WILL BE REDUCED; WITH A MAXIMUM FACTOR OF 16
           YSPMIN = YSPMIN/YSPMAX
           DENO  = 1. + 15* YSPMIN

           CELLTI(ICELL) = CELLTI(ICELL)/ABS(DENO)
185        CONTINUE
C
C       --------------------------------
C       GLOBAL MINIMUM/MAXIMUM TIME STEPS
C       --------------------------------
C
190     DO 200 JCELL = 1, NCELA2

C          FIND THE ACTUAL CELL NUMBER
           ICELL = ICELA2(JCELL)

           DTMNTI = MIN ( DTMNTI, CELLTI(ICELL) )
           DTMAX  = MAX ( DTMAX , CELLTI(ICELL) )
200     CONTINUE
C
C       ------------------
C       BOUNDARY NODE CELLS
C       ------------------
C
C       SKIP THE NEXT SECTION IF NO CHARACTERISTIC B.C'S ARE USED
C
        IF (NCRSG2 .EQ. 0) GO TO 250
C
C       RESET THE CELL TIME STEPS FOR THE BOUNDARY CELLS INVOLVED
C       WITH CHARACTERISTIC BOUNDARY CONDITIONS.  FIRST INITIALIZE
C       NUMBER OF CELLS IN EACH BOUNDARY TYPE AND THEIR TIME STEPS
C

        DO 210 N = 1, MMAXTI
          NCEL(N)   = 0
          CELLBN(N) = 1000.
210     CONTINUE
C
C       COLLECT THE NODES WITH SPECIFIC CHARACTERISTICS
C       ONLY TYPES OF KIND 4, 5, 6 WILL BE CONSIDERED
C
        DO 220 IBND = 1, NBNDG2
          ITYPE = IBNDG2(5,IBND)
c         N     = ITYPE - 2
          N     = ITYPE - 3
          IF (N .GT. 0 .AND. N .LE. MAXCHR) THEN
            NCEL(N)           = NCEL(N) + 1
            ICELTT(N,NCEL(N)) = IBND
            TIMCL1            = CELLTI(IBNDG2(2,IBND))
            IF (IBNDG2(3,IBND) .NE. 0) THEN
               TIMCL2 = CELLTI(IBNDG2(3,IBND))
            ELSE
               TIMCL2 = 1000.
            ENDIF
```

```
            CELLBN(N) = MIN (CELLBN(N), TIMCL1, TIMCL2)
          ENDIF
220     CONTINUE
C
C       NOW RESET THE CELLS ASSOCIATED WITH BOUNDARY NODES TO HAVE
C       THE MINIMUM TIME STEP OVER A PARTICULAR TYPE
C
        DO 240 N = 1, MAXCHR
          DO 230 JBND = 1, NCEL(N)
             IBND                    = ICELTT(N,JBND)
             CELLTI(IBNDG2(2,IBND)) = CELLBN(N)
C            CELLTI(IBNDG2(3,IBND)) = CELLBN(N)
230       CONTINUE
240     CONTINUE
C
C       ----------------------
C       MAXIMUM TEMPORAL LEVEL
C       ----------------------
C
C       COMPUTE THE MAXIMUM TEMPORAL LEVEL OF CELLS

250     AKMAX  = DTMAX/DTMNTI
        KMAX   = 10000
        IF (AKMAX .LT. 1.E4) KMAX = NINT(AKMAX)
        ZKK    = KMAX
        ZZ     = LOG (ZKK) / ZBASLG
        N      = INT (ZZ)
        NMAXTI = MIN (N,NGIVTI,MMAXTI)
C
C       -----------------------------
C       TEMPORAL LEVEL CELLS GROUPINGS
C       -----------------------------
C
C       INITIALIZE THE NUMBER OF CELLS IN EACH LEVEL

        NCELO  = 0

CVD$    NOVECTOR
CVD$    NOLSTVAL
        DO 260 N = 1, MMAXTI
          NCEL(N) = 0
260     CONTINUE

C       REASSIGN THE CELL TIME STEPS AS INTEGRAL MULTIPLES OF 2

CVD$    NOLSTVAL
CVD$    NODEPCHK
        DO 270 JCELL     = 1, NCELA2
          ICELL          = ICELA2(JCELL)
          K              = 10000
          AK             = CELLTI(ICELL)/DTMNTI
          IF (AK .LT. 1.E4) K = NINT(AK)
          ZKK            = K
          ZZ             = LOG (ZKK) / ZBASLG
          N              = INT (ZZ)
          N              = MIN (N,NMAXTI)
          IP             = 2**N
```

```
          CELLTI(ICELL) = IP*DTMNTI
270       CONTINUE

C         SEE IF YOU WANT TO TRANSLATE NODITS, OR WANT TO LIMIT THE
C         CELL TIME STEPS BY FACTORS OF FOUR AT MOST

          IF (NMAXTI .GT. 1 .and. kadphr .ne. 0) CALL NODIT2

          DO 280 JCELL    = 1, NCELA2
            ICELL         = ICELA2(JCELL)
            AK            = CELLTI(ICELL)/DTMNTI
            K             = NINT(AK)
            ZKK           = K
            ZZ            = LOG (ZKK) / ZBASLG
            N             = INT (ZZ)
            N             = MIN (N,NMAXTI)

            IF (N .EQ. 0) THEN
               NCELO         = NCELO + 1
               ICELTI(NCELO) = ICELL
            ELSE
               NCEL(N)           = NCEL(N) + 1
               ICELTT(N,NCEL(N)) = ICELL
            ENDIF
280       CONTINUE

C         NOW SET UP THE POINTER SYSTEM FOR TEMPORAL ADAPTATION

          ILVLTI(1,0) = 1
          ILVLTI(2,0) = NCELO

CVD$      NOVECTOR
CVD$      NOLSTVAL
          DO 300 N = 1, NMAXTI
            NCELT       = NCEL(N)
            ILVLTI(1,N) = ILVLTI(2,N-1) + 1
            ILVLTI(2,N) = ILVLTI(1,N  ) + NCELT - 1
            DO 290 JCELL = 1, NCELT
              NCELO         = NCELO + 1
              ICELTI(NCELO) = ICELTT(N,JCELL)
290         CONTINUE
300       CONTINUE

          RETURN
          END
```

E2TIM0

```
          SUBROUTINE E2TIM0

          INCLUDE '[.INC] PRECIS.INC/LIST'
          INCLUDE '[.INC] PARMV2.INC/LIST'
          INCLUDE '[.INC] A2COMN.INC/LIST'
```

```
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] JACOMN.INC/LIST'
      INCLUDE '[.INC] TICOMN.INC/LIST'

      DIMENSION NCEL(MMAXTI), ICELTT(MMAXTI,MCELG2), CELLBN(MMAXTI)

C*********************************************************************

C     THIS SUBROUTINE STEPS THROUGH EACH CELL ON THIS LEVEL AND
C     COMPUTES THE CELL TIME STEP AS A FIRST STEP.  THE CELL TIME-
C     STEPS ARE REASSIGNED AS MULTIPLES OF INTEGRAL POWERS OF 2
C     TIMES THE GLOBAL MIMIMUM TIME-STEP IF ATLEAST A FACTOR OF
C     2 EXISTS BETWEEN GLOBAL MIMIMUM AND GLOBAL MAXIMUM VALUES.

C*********************************************************************

C     INITIALIZE QUANTITIES

      DTMNTI = 1000.
      DTMAX  = 0.
      GAMMA  = 1.66
      MAXCHR = 4

C     STEP THROUGH ALL THE CEWIC CELLS AND FIND CELL TIMESTEPS

      DO 20 JCELL = 1, NCELA2

C        ----------------------
C        CELL/NODE DETERMINATION
C        ----------------------

C        FIND THE ACTUAL CELL NUMBER
         ICELL = ICELA2(JCELL)

C        SET UP NODE POINTERS FOR THIS CELL

         KSW = ICELG2(2,ICELL)
         KSE = ICELG2(4,ICELL)
         KNE = ICELG2(6,ICELL)
         KNW = ICELG2(8,ICELL)

C        --------
C        GEOMETRY
C        --------
C
C        GEOMETRY OF ALL CELL CORNERS
C
         XSW = GEOMG2(1,KSW)
         YSW = GEOMG2(2,KSW)
C
         XSE = GEOMG2(1,KSE)
         YSE = GEOMG2(2,KSE)
C
         XNE = GEOMG2(1,KNE)
         YNE = GEOMG2(2,KNE)
```

```
C
      XNW = GEOMG2(1,KNW)
      YNW = GEOMG2(2,KNW)

C     COMPUTE THE PROJECTIONS OF CELL FACES AND THE SIZE OF THE CELL

      DXEW = 0.5*(XNE + XSE - XNW - XSW)
      DYEW = 0.5*(YNE + YSE - YNW - YSW)
      DXNS = 0.5*(XNW + XNE - XSW - XSE)
      DYNS = 0.5*(YNW + YNE - YSW - YSE)

      DVOL = 0.5*( (XSE-XNW)*(YNE-YSW) - (YSE-YNW)*(XNE-XSW) )
C
C     -----------------
C     CELL CENTER VALUES
C     -----------------
C
C     DETERMINE THE DEPENDENT VARIABLES AT THE CENTER OF THE CELL
C     COMPUTE FIRST THE DEPENDENT VARIABLES AT THE FACES

      DO 10 IQ = 1, NEQNFL

          DPENSW = DPENG2(IQ,KSW)
          DPENSE = DPENG2(IQ,KSE)
          DPENNE = DPENG2(IQ,KNE)
          DPENNW = DPENG2(IQ,KNW)

C         AVERAGE VALUES AT THE CENTER

          DPENJA(IQ) = 0.25*( DPENSW + DPENSE + DPENNE + DPENNW )

10    CONTINUE

C     GET THE PRESSURE FOR THE CELL

      PRESSW = PRESG2(KSW)
      PRESSE = PRESG2(KSE)
      PRESNE = PRESG2(KNE)
      PRESNW = PRESG2(KNW)

C     AVERAGE VALUE AT THE CENTER

      PRESPR = 0.25*( PRESSW + PRESSE + PRESNE + PRESNW )
C
C     -------------
C     CFL CONDITION
C     -------------
C
C     COMPUTE THE VELOCITY COMPONENTS AND SPEED OF SOUND

      UCOMPC = DPENJA(2)/DPENJA(1)
      VCOMPC = DPENJA(3)/DPENJA(1)
      SOUND  = GAMMA*PRESPR/DPENJA(1)
      SOUND  = SQRT(SOUND)

C     COMPUTE AVERAGE DISTANCES
```

```
      DISTEW = SQRT(DXEW*DXEW + DYEW*DYEW )
      DISTNS = SQRT(DXNS*DXNS + DYNS*DYNS )

C     COMPUTE THE CFL CONDITION IN THE TWO DIRECTIONS

      DTEW   = ABS(UCOMPC*DYEW - VCOMPC*DXEW) + SOUND*DISTEW
      DTNS   = ABS(UCOMPC*DYNS - VCOMPC*DXNS) + SOUND*DISTNS
      EIGEN  = MAX(DTEW,DTNS)
      DTN    = CFLNTI*DVOL/EIGEN

      CELLTI(ICELL) = DTN*FCTRTI

      IF (DTN .LE. 0.) THEN
        ZER1 = ICELL
        ZER2 = DVOL
        CALL ERRORM (4,'E2TIMO','ICELL ',ZER1,'DVOL  ',ZER2,JPRINT,
     1          'CELL TIME STEP IS ZERO')
      ENDIF
C
C     ------------------
C     TEMPORAL RESOLUTION
C     ------------------
C
C     SEE IF YOU WANT TO SKIP THE TEMPORAL RESOLUTION

      IF (KADPTI .NE. 0) THEN
C        COMPUTE THE SOURCE TERM AT THE CELL CENTER
         CALL E2SOUR
         BIGWCE = BIGWJA(KADPTI)
C        NOW COMPUTE THE FLUX TERMS AT THE FOUR CORNER NODES
         UCOMPSW = DPENG2(2,KSW)/DPENG2(1,KSW)
         UCOMPSE = DPENG2(2,KSE)/DPENG2(1,KSE)
         UCOMPNE = DPENG2(2,KNE)/DPENG2(1,KNE)
         UCOMPNW = DPENG2(2,KNW)/DPENG2(1,KNW)
         VCOMPSW = DPENG2(3,KSW)/DPENG2(1,KSW)
         VCOMPSE = DPENG2(3,KSE)/DPENG2(1,KSE)
         VCOMPNE = DPENG2(3,KNE)/DPENG2(1,KNE)
         VCOMPNW = DPENG2(3,KNW)/DPENG2(1,KNW)
         BIGFSW  = UCOMPSW*DPENG2(KADPTI,KSW)
         BIGFSE  = UCOMPSE*DPENG2(KADPTI,KSE)
         BIGFNE  = UCOMPNE*DPENG2(KADPTI,KNE)
         BIGFNW  = UCOMPNW*DPENG2(KADPTI,KNW)

         BIGGSW  = VCOMPSW*DPENG2(KADPTI,KSW)
         BIGGSE  = VCOMPSE*DPENG2(KADPTI,KSE)
         BIGGNE  = VCOMPNE*DPENG2(KADPTI,KNE)
         BIGGNW  = VCOMPNW*DPENG2(KADPTI,KNW)

         IF (KADPTI .EQ. 2) THEN
            BIGFSW  = BIGFSW + PRESG2(KSW)
            BIGFSE  = BIGFSE + PRESG2(KSE)
            BIGFNE  = BIGFNE + PRESG2(KNE)
            BIGFNW  = BIGFNW + PRESG2(KNW)
         ENDIF

         IF (KADPTI .EQ. 3) THEN
            BIGGSW  = BIGGSW + PRESG2(KSW)
```

```
-       BIGGSE  = BIGGSE + PRESG2(KSE)
-       BIGGNE  = BIGGNE + PRESG2(KNE)
        BIGGNW  = BIGGNW + PRESG2(KNW)
      ENDIF

      IF (KADPTI .EQ. 4) THEN
        BIGFSW  = BIGFSW + PRESG2(KSW)*UCOMPSW
        BIGFSE  = BIGFSE + PRESG2(KSE)*UCOMPSE
        BIGFNE  = BIGFNE + PRESG2(KNE)*UCOMPNE
        BIGFNW  = BIGFNW + PRESG2(KNW)*UCOMPNW
        BIGGSW  = BIGGSW + PRESG2(KSW)*VCOMPSW
        BIGGSE  = BIGGSE + PRESG2(KSE)*VCOMPSE
        BIGGNE  = BIGGNE + PRESG2(KNE)*VCOMPNE
        BIGGNW  = BIGGNW + PRESG2(KNW)*VCOMPNW
      ENDIF

C     COMPUTE THE FIRST ORDER CHANGE (/CELLTI)
      DENO  = (BIGFSW-BIGFNE)*(YNW-YSE) +
     1        (BIGFNW-BIGFSE)*(YNE-YSW) +
     2        (BIGGSW-BIGGNE)*(XSE-XNW) +
     3        (BIGGNW-BIGGSE)*(XSW-XNE)

      DENO  = BIGWCE + 0.5*DENO/DVOL
      UCELL = DPENJA(KADPTI)

      DTR   = 1000.

      IF (DENO .NE. 0.) THEN
        DTR   = EPS1TI*(EPSOTI+UCELL)/ABS(DENO)
      ENDIF

      IF (DTR .LE. 0.) THEN
        ZER1 = ICELL
        ZER2 = DVOL
        CALL ERRORM (4,'E2TIMO','ICELL ',ZER1,'DVOL  ',ZER2,
     1     JPRINT,'CELL TIME STEP IS ZERO')
      ENDIF

      CELLTI(ICELL) = MIN (DTN, DTR)

    ENDIF

C   COMPUTE THE GLOBAL MINIMUM AND MAXIMUM TIME STEPS

    DTMNTI = MIN ( DTMNTI, CELLTI(ICELL) )
    DTMAX  = MAX ( DTMAX , CELLTI(ICELL) )

C   GO BACK FOR NEXT CEWIC CELL
C
20  CONTINUE
C
C   ------------------
C   BOUNDARY NODE CELLS
C   ------------------
C
C   SKIP THE NEXT SECTION IF NO CHARACTERISTIC B.C'S ARE USED
C
```

```
      IF (NCRSG2 .EQ. 0) GO TO 70
C
C     RESET THE CELL TIME STEPS FOR THE BOUNDARY CELLS INVOLVED
C     WITH CHARACTERISTIC BOUNDARY CONDITIONS.  FIRST INITIALIZE
C     NUMBER OF CELLS IN EACH BOUNDARY TYPE AND THEIR TIME STEPS
C
      DO 30 N = 1, MMAXTI
        NCEL(N)   = 0
        CELLBN(N) = 1000.
30    CONTINUE
C
C     COLLECT THE NODES WITH SPECIFIC CHARACTERISTICS
C     ONLY TYPES OF KIND 4, 5, 6 WILL BE CONSIDERED
C
      DO 40 IBND = 1, NBNDG2
        ITYPE = IBNDG2(5,IBND)
        N     = ITYPE - 2
C       N     = ITYPE - 3
        IF (N .GT. 0 .AND. N .LE. MAXCHR) THEN
          NCEL(N)           = NCEL(N) + 1
          ICELTT(N,NCEL(N)) = IBND
          TIMCL1            = CELLTI(IBNDG2(2,IBND))
          IF (IBNDG2(3,IBND) .NE. 0) THEN
             TIMCL2 = CELLTI(IBNDG2(3,IBND))
          ELSE
             TIMCL2 = 1000.
          ENDIF
          CELLBN(N) = MIN (CELLBN(N), TIMCL1, TIMCL2)
        ENDIF
40    CONTINUE
C
C     NOW RESET THE CELLS ASSOCIATED WITH BOUNDARY NODES TO HAVE
C     THE MINIMUM TIME STEP OVER A PARTICULAR TYPE
C
      DO 60 N = 1, MAXCHR
        DO 50 JBND = 1, NCEL(N)
           IBND                    = ICELTT(N,JBND)
           CELLTI(IBNDG2(2,IBND)) = CELLBN(N)
           CELLTI(IBNDG2(3,IBND)) = CELLBN(N)
50      CONTINUE
60    CONTINUE
C
C     ----------------------
C     MAXIMUM TEMPORAL LEVEL
C     ----------------------
C
C     COMPUTE THE MAXIMUM TEMPORAL LEVEL OF CELLS

70    AKMAX  = DTMAX/DTMNTI
      KMAX   = 10000
      IF (AKMAX .LT. 1.E4) KMAX = NINT(AKMAX)
      IF(KMAX .LE. 0) THEN
         ZER1 = KMAX
         ZER2 = DTMAX
         CALL ERRORM (5,'E2TIMO','KMAX  ',ZER1,'DTMAX ',ZER2,JPRINT,
     1          'ERROR IN MAXIMUM TEMPORAL LEVEL CALCULATION')
```

```
      ENDIF

      NMAXTI = IBASE2(KMAX,MMAXTI)
      NMAXTI = MIN(NGIVTI,NMAXTI)
C
C     ------------------------------
C     TEMPORAL LEVEL CELLS GROUPINGS
C     ------------------------------
C
C     INITIALIZE THE NUMBER OF CELLS IN EACH LEVEL

      NCELO  = 0
      DO 80 N = 1, MMAXTI
        NCEL(N) = 0
80    CONTINUE

C     REASSIGN THE CELL TIME STEPS AS INTEGRAL MULTIPLES OF 2

      DO 90 JCELL     = 1, NCELA2
        ICELL         = ICELA2(JCELL)
        K             = 10000
        AK            = CELLTI(ICELL)/DTMNTI
        IF (AK .LT. 1.E4) K = NINT(AK)
        N             = IBASE2(K,NMAXTI)
        IP            = 2**N
        CELLTI(ICELL) = IP*DTMNTI
90    CONTINUE

C     SEE IF YOU WANT TO TRANSLATE NODITS, OR WANT TO LIMIT THE
C     CELL TIME STEPS BY FACTORS OF FOUR AT MOST
C     **** NODIT2 DOESNOT TRANSLATE NODITS IN 2-D YET ****

      IF (NMAXTI .GT. 0) CALL NODIT2

      DO 100 JCELL    = 1, NCELA2
        ICELL         = ICELA2(JCELL)
        AK            = CELLTI(ICELL)/DTMNTI
        K             = NINT(AK)
        N             = IBASE2(K,NMAXTI)
        IF (N .EQ. 0) THEN
           NCELO         = NCELO + 1
           ICELTI(NCELO) = ICELL
        ELSE
           NCEL(N)           = NCEL(N) + 1
           ICELTT(N,NCEL(N)) = ICELL
        ENDIF
100   CONTINUE

C     NOW SET UP THE POINTER SYSTEM FOR TEMPORAL ADAPTATION

      ILVLTI(1,0) = 1
      ILVLTI(2,0) = NCELO

      DO 120 N = 1, NMAXTI
        NCELT       = NCEL(N)
        ILVLTI(1,N) = ILVLTI(2,N-1) + 1
        ILVLTI(2,N) = ILVLTI(1,N  ) + NCELT - 1
```

```
          DO 110 JCELL = 1, NCELT
            NCELO         = NCELO + 1
            ICELTI(NCELO) = ICELTT(N,JCELL)
110       CONTINUE
120     CONTINUE

        IF (IDBGE2 .GT. 1000 .AND. NMAXTI .GT. 0) THEN
           CALL TIPRN2 (JDEBUG)
        ENDIF

        RETURN
        END
```

E2TIMC

```
        SUBROUTINE E2TIMC

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] A2COMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
        INCLUDE '[.INC] HEXCOD.INC     '
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        INCLUDE '[.INC] TICOMN.INC/LIST'
        DIMENSION NCEL(MMAXTI), ICELTT(MMAXTI,MCELG2), CELLBN(MMAXTI)

C***********************************************************************

C       THIS SUBROUTINE SETS THE TIME-STEP OF EACH BASE LEVEL CELL AS A
C       CONSTANT; WHEREAS HIGHER CELLS CELLS HAVE CORRESPONDINGLY
C       LESSER TIME-STEPS.

C***********************************************************************

C       INITIALIZE QUANTITIES

        MAXLEV = 0
        MAXCHR = 3

C       STEP THROUGH ALL THE CEWIC CELLS AND FIND CELL TIMESTEPS

        DO 20 JCELL = 1, NCELA2

C          FIND THE ACTUAL CELL NUMBER
           ICELL = ICELA2(JCELL)

C          FIND THE LEVEL LEVELG OF THE GIVEN CELL
           KX     = KAUXG2(LCELL)
           K5LEVG = IAND(KX,KU000F)
           LEVELG = ISHFT(K5LEVG,-16)
           IDENO  = 1
           IF (LEVELG .GT. 0) IDENO = 2**LEVELG
           MAXLEV = MAX (MAXLEV, LEVELG)
```

```
            CELLTI(ICELL) = DTCNTI/IDENO*FCTRTI

C           GO BACK FOR NEXT CEWIC CELL
C
20        CONTINUE

C         COMPUTE THE GLOBAL MINIMUM AND MAXIMUM TIME STEPS

          DTMNTI = DTCNTI/(2**MAXLEV)
          DTMAX  = DTCNTI
C
C         ------------------
C         BOUNDARY NODE CELLS
C         ------------------
C
C         SKIP THE NEXT SECTION IF NO CHARACTERISTIC B.C'S ARE USED
C
          IF (NCRSG2 .EQ. 0) GO TO 70
C
C         RESET THE CELL TIME STEPS FOR THE BOUNDARY CELLS INVOLVED
C         WITH CHARACTERISTIC BOUNDARY CONDITIONS.  FIRST INITIALIZE
C         NUMBER OF CELLS IN EACH BOUNDARY TYPE AND THEIR TIME STEPS
C
          DO 30 N = 1, MMAXTI
            NCEL(N)   = 0
            CELLBN(N) = 1000.
30        CONTINUE
C
C         COLLECT THE NODES WITH SPECIFIC CHARACTERISTICS
C         ONLY TYPES OF KIND 4, 5, 6 WILL BE CONSIDERED
C
          DO 40 IBND = 1, NBNDG2
            ITYPE = IBNDG2(5,IBND)
            N     = ITYPE - 3
            IF (N .GT. 0 .AND. N .LE. MAXCHR) THEN
              NCEL(N)           = NCEL(N) + 1
              ICELTT(N,NCEL(N)) = IBND
              TIMCL1            = CELLTI(IBNDG2(2,IBND))
              IF (IBNDG2(3,IBND) .NE. 0) THEN
                 TIMCL2 = CELLTI(IBNDG2(3,IBND))
              ELSE
                 TIMCL2 = 1000.
              ENDIF
              CELLBN(N) = MIN (CELLBN(N), TIMCL1, TIMCL2)
            ENDIF
40        CONTINUE
C
C         NOW RESET THE CELLS ASSOCIATED WITH BOUNDARY NODES TO HAVE
C         THE MINIMUM TIME STEP OVER A PARTICULAR TYPE
C
          DO 60 N = 1, MAXCHR
            DO 50 JBND = 1, NCEL(N)
               IBND                    = ICELTT(N,JBND)
               CELLTI(IBNDG2(2,IBND)) = CELLBN(N)
50          CONTINUE
60        CONTINUE
C
```

```
C       ---------------------
C       MAXIMUM TEMPORAL LEVEL
C       ---------------------
C
C       COMPUTE THE MAXIMUM TEMPORAL LEVEL OF CELLS

70      AKMAX  = DTMAX/DTMNTI
        KMAX   = NINT(AKMAX)

        NMAXTI = IBASE2(KMAX,MMAXTI)
        NMAXTI = MIN(NGIVTI,NMAXTI)
C
C       INITIALIZE THE NUMBER OF CELLS IN EACH LEVEL
C
        NCELO  = 0
        DO 80 N = 1, MMAXTI
          NCEL(N) = 0
80      CONTINUE

        DO 100 JCELL    = 1, NCELA2
          ICELL         = ICELA2(JCELL)
          AK            = CELLTI(ICELL)/DTMNTI
          K             = NINT(AK)
          N             = IBASE2(K,NMAXTI)
          IF (N .EQ. 0) THEN
             NCELO         = NCELO + 1
             ICELTI(NCELO) = ICELL
          ELSE
             NCEL(N)           = NCEL(N) + 1
             ICELTT(N,NCEL(N)) = ICELL
          ENDIF
100     CONTINUE
C
C       NOW SET UP THE POINTER SYSTEM FOR TEMPORAL ADAPTATION

        ILVLTI(1,0) = 1
        ILVLTI(2,0) = NCELO

        DO 120 N = 1, NMAXTI
          NCELT       = NCEL(N)
          ILVLTI(1,N) = ILVLTI(2,N-1) + 1
          ILVLTI(2,N) = ILVLTI(1,N  ) + NCELT - 1
          DO 110 JCELL = 1, NCELT
            NCELO         = NCELO + 1
            ICELTI(NCELO) = ICELTT(N,JCELL)
110       CONTINUE
120     CONTINUE

        RETURN
        END
```

E2UPDF

```
      SUBROUTINE E2UPDO (ITGL)
C                E2UPDF

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'PRCOMN.INC'

C*********************************************************************

C     THIS SUBROUTINE UPDATES THE DEPENDENT VARIABLES AT EACH NODE
C     ASSOCIATED WITH A CELL ON THIS AND ALL FINER CELLS

C*********************************************************************

C     LOOP OVER ALL HANGING NODES AND INTERPOLATE

      DO 20 J  = 1, NEQNFL
         DO 10 JNODE = 1, NHNGA2
            INODE           = MRKCA2(JNODE)
            ICOR1           = NINT(WORKA2(JNODE))
            ICOR2           = NINT(CHNGA2(JNODE))
            DPENG2(J,INODE) = 0.5*(DPENG2(J,ICOR1) + DPENG2(J,ICOR2))
            CHNGE2(J,INODE) = 0.
10       CONTINUE
20    CONTINUE
C
C     A SIMPLER MODEL IS USED FOR PRESSURE AND TEMPERATURE TO
C     AVOID EXPANSIVE CALCULATIONS.

      DO 30 JNODE = 1, NHNGA2
         INODE         = MRKCA2(JNODE)
         ICOR1         = NINT(WORKA2(JNODE))
         ICOR2         = NINT(CHNGA2(JNODE))
         PRESG2(INODE) = 0.5*(PRESG2(ICOR1) + PRESG2(ICOR2))
         TEMPG2(INODE) = 0.5*(TEMPG2(ICOR1) + TEMPG2(ICOR2))
30    CONTINUE
C
C
C     LOOP OVER ALL THE NODES AT THIS LEVEL AND UPDATE THEM

      DO 50 J  = 1, NEQNFL
         DO 40 JNODE = ILVLA2(1,ITGL), ILVLA2(2,ITGL)
            INODE           = MRKDA2(JNODE)
            DPENG2(J,INODE) = DPENG2(J,INODE) + CHNGE2(J,INODE)
            CHNGE2(J,INODE) = 0.
40       CONTINUE
50    CONTINUE
C
C     NOW COMPUTE AND CORRECT THE PRIMITIVE VARIABLES
```

```
C
      DO 90 JNODE = ILVLA2(1,ITGL), ILVLA2(2,ITGL)
C
          INODE  = MRKDA2(JNODE)
          RHORPR = DPENG2(1,INODE)
          UCOMPR = DPENG2(2,INODE)/RHORPR
          VCOMPR = DPENG2(3,INODE)/RHORPR
          BEPSPR = DPENG2(4,INODE)
          BEU    = BEPSPR/RHORPR
          VELO2U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C         COMPUTE THE DIMENSIONAL QUANTITIES
C
          BE     = FMREFL*BEU
          VELO2  = FMREFL*VELO2U

C         COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

          SUMY   = 0.

          DO 60 IS = 1, NEQSCH

              JS         = NEQBAS + IS
              YSPEPR(IS) = DPENG2(JS,INODE)/DPENG2(1,INODE)

              IF (YSPEPR(IS) .LT. 0.) THEN
                YSPEPR(IS)       = 0.
                DPENG2(JS,INODE) = 0.
              ENDIF

              IF (YSPEPR(IS) .GT. YMAXCH(IS)) THEN
                YSPEPR(IS)       = YMAXCH(IS)
                DPENG2(JS,INODE) = YMAXCH(IS)*DPENG2(1,INODE)
              ENDIF

              SUMY       = SUMY + YSPEPR(IS)
60        CONTINUE

          YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
C         YSPEPR(NEQSCH+1) = ABS(1. - SUMY - YNRTCH)
          IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
          IF (YSPEPR(NEQSCH+1) .GT. YMAXCH(NEQSCH+1))
     1         YSPEPR(NEQSCH+1) = YMAXCH(NEQSCH+1)

          IF (KROGER .EQ. 1) THEN
C
             TEMPD = TEMPG2(INODE)*TREFFL
             IF (TEMPD .LT. TRIGCH) GOTO 70
C
C            USE THE FIRST REACTION AS EQUILIBRIUM REACTION, IF THE
C            CONCENTRATIONS ARE FAR AWAY FROM EQUILIBRIUM
C
             AKEQ   = 117.31948*EXP(-8992./TEMPD)
             YOHEQ  = SQRT(YSPEPR(3)*YSPEPR(1)*AKEQ)
             DELTAY = YOHEQ-YSPEPR(2)
C
             IF (    DELTAY  .GT. 0.01*YMAXCH(2) .AND.
```

```
     1              YSPEPR(4) .LT. 0.50*YMAXCH(4)) THEN
            -     DELTAY          = 0.5*DELTAY*RAMWCH(2)
            -     YO2EQ           = YSPEPR(1) - AMWTCH(1)*DELTAY
                  YH2EQ           = YSPEPR(3) - AMWTCH(3)*DELTAY
C                 RESET THE DEPENDENT VARIABLES
                  DPENG2(5,INODE) = RHORPR*YO2EQ
                  DPENG2(6,INODE) = RHORPR*YOHEQ
                  DPENG2(7,INODE) = RHORPR*YH2EQ
                  YSPEPR(1)       = YO2EQ
                  YSPEPR(2)       = YOHEQ
                  YSPEPR(3)       = YH2EQ
               ENDIF
C
            ENDIF
C
70          SYSHFS = 0.
            SYSCPS = 0.
            SYSBMS = 0.
            BIGAM  = 0.
C
C           COMPUTE THE TEMPERATURE IN DEGREE K
C
            DO 80 IS = 1, NSPECH
                SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
                SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
                SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
                BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
80          CONTINUE

            BIGBM   = SYSCPS - UGASFL*SYSBMS
            BIGCM   = BE - 0.5*VELO2 - SYSHFS + TREFCH*SYSCPS
     1                           + 0.5*TREFCH*TREFCH*BIGAM

            IF (BIGAM .LT. 1.E-10) THEN
               TEMPD  = BIGCM/BIGBM
            ELSE
               DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
               TEMPD  = ( SQRT(DISCRI)-BIGBM )/BIGAM
            ENDIF
C
C           NORMALIZE THE TEMPERATURE
C
            TEMPPR = TEMPD/TREFFL
C
C           COMPUTE THE DIMENSIONLESS PRESSURE
C
            PRESPR  = RHORPR*TEMPPR*AMWTFL*SYSBMS
C           IF (PRESPR .LE. 0.) CALL CHKPR2(INODE)
C
C           SAVE THE PRESSURE AND TEMPERATURE AT THE NODE
C
            PRESG2(INODE) = PRESPR
            TEMPG2(INODE) = TEMPPR

90       CONTINUE
C
         RETURN
```

```
      END
```

E2UPDU

```
      SUBROUTINE E2UPDO (ITGL)
C                E2UPDU

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.INC     '
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] TICOMN.INC/LIST'
      DIMENSION MARK(0:MNODG2)

C*********************************************************************

C     THIS SUBROUTINE UPDATES THE DEPENDENT VARIABLES AT EACH NODE
C     ASSOCIATED WITH A CELL ON THIS AND ALL FINER CELLS

C*********************************************************************

C     MARK ALL THE NODES FOR THE CASE FOR SUBSEQUENT UPDATING

      MARK(0) = 0
      DO 10 INODE = 1, NNODG2
         MARK(INODE) = 1
10    CONTINUE

C     LOOP OVER ALL THE CELLS AT THIS LEVEL AND UPDATE THE
C     CORRESPONDING NODES

      DO 100 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C        NODE/CELL ASSIGNMENTS

         ICELL = ICELTI (  JCELL)
         KSW   = ICELG2 (2,ICELL)
         KS    = ICELG2 (3,ICELL)
         KSE   = ICELG2 (4,ICELL)
         KE    = ICELG2 (5,ICELL)
         KNE   = ICELG2 (6,ICELL)
         KN    = ICELG2 (7,ICELL)
         KNW   = ICELG2 (8,ICELL)
         KW    = ICELG2 (9,ICELL)

C        CHECK IF UPDATING IS TO BE DONE AT THE SOUTHWESTERN NODE

         IF ( MARK(KSW) .NE. 0 ) THEN
            MARK(KSW) = 0
            DO 20 J  = 1, NEQNFL
              DPENG2(J,KSW) = DPENG2(J,KSW) + CHNGE2(J,KSW)
```

```
                CHNGE2(J,KSW) = 0.
20          -   CONTINUE
            -   CALL E2PRMT (KSW, 1)
            ENDIF

C           CHECK IF UPDATING IS TO BE DONE AT THE SOUTHERN EDGE

            IF ( MARK(KS) .NE. 0 ) THEN
                MARK(KS) = 0
                DO 30 J  = 1, NEQNFL
                  DPENG2(J,KS) = DPENG2(J,KS) + CHNGE2(J,KS)
                  CHNGE2(J,KS) = 0.
CTEST
c                 DPENG2(J,KS) = 0.5*(DPENG2(J,KSW) + DPENG2(J,KSE))
CTEST
30              CONTINUE
                CALL E2PRMT (KS, 1)
            ENDIF

C           CHECK IF UPDATING IS TO BE DONE AT THE SOUTHEASTERN NODE

            IF ( MARK(KSE) .NE. 0 ) THEN
                MARK(KSE) = 0
                DO 40 J  = 1, NEQNFL
                  DPENG2(J,KSE) = DPENG2(J,KSE) + CHNGE2(J,KSE)
                  CHNGE2(J,KSE) = 0.
40              CONTINUE
                CALL E2PRMT (KSE, 1)
            ENDIF

C           CHECK IF UPDATING IS TO BE DONE AT THE EASTERN EDGE

            IF ( MARK(KE) .NE. 0 ) THEN
                MARK(KE) = 0
                DO 50 J  = 1, NEQNFL
                  DPENG2(J,KE) = DPENG2(J,KE) + CHNGE2(J,KE)
                  CHNGE2(J,KE) = 0.
CTEST
c                 DPENG2(J,KE) = 0.5*(DPENG2(J,KSE) + DPENG2(J,KNE))
CTEST
50              CONTINUE
                CALL E2PRMT (KE, 1)
            ENDIF

C           CHECK IF UPDATING IS TO BE DONE AT THE NORTHEASTERN NODE

            IF ( MARK(KNE) .NE. 0 ) THEN
                MARK(KNE) = 0
                DO 60 J  = 1, NEQNFL
                  DPENG2(J,KNE) = DPENG2(J,KNE) + CHNGE2(J,KNE)
                  CHNGE2(J,KNE) = 0.
60              CONTINUE
                CALL E2PRMT (KNE, 1)
            ENDIF

C           CHECK IF UPDATING IS TO BE DONE AT THE NORTHERN EDGE
```

```
            IF ( MARK(KN) .NE. 0 ) THEN
            -   MARK(KN) = 0
            -   DO 70 J  = 1, NEQNFL
                  DPENG2(J,KN) = DPENG2(J,KN) + CHNGE2(J,KN)
                  CHNGE2(J,KN) = 0.
CTEST
c                 DPENG2(J,KN) = 0.5*(DPENG2(J,KNW) + DPENG2(J,KNE))
CTEST
70              CONTINUE
                CALL E2PRMT (KN, 1)
            ENDIF

C           CHECK IF UPDATING IS TO BE DONE AT THE NORTHWESTERN NODE

            IF ( MARK(KNW) .NE. 0 ) THEN
                MARK(KNW) = 0
                DO 80 J  = 1, NEQNFL
                  DPENG2(J,KNW) = DPENG2(J,KNW) + CHNGE2(J,KNW)
                  CHNGE2(J,KNW) = 0.
80              CONTINUE
                CALL E2PRMT (KNW, 1)
            ENDIF

C           CHECK IF UPDATING IS TO BE DONE AT THE WESTERN EDGE

            IF ( MARK(KW) .NE. 0 ) THEN
                MARK(KW) = 0
                DO 90 J  = 1, NEQNFL
                  DPENG2(J,KW) = DPENG2(J,KW) + CHNGE2(J,KW)
                  CHNGE2(J,KW) = 0.
CTEST
c                 DPENG2(J,KW) = 0.5*(DPENG2(J,KSW) + DPENG2(J,KNW))
CTEST
90              CONTINUE
                CALL E2PRMT (KW, 1)
            ENDIF

100     CONTINUE
C
        RETURN
        END
```

E2UPD0

```
        SUBROUTINE E2UPD0 (ITGL)

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] E2COMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
        INCLUDE '[.INC] HEXCOD.INC     '
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        INCLUDE '[.INC] TICOMN.INC/LIST'
```

```
      DIMENSION MARK(0:MNODG2)
      LOGICAL   IWRITE
      DATA      KOUNT /0/

C*********************************************************************

C       THIS SUBROUTINE UPDATES THE DEPENDENT VARIABLES AT EACH NODE
C       ASSOCIATED WITH A CELL ON THIS AND ALL FINER CELLS

C*********************************************************************

C       CHECK IF DEBUG PRINT IS NEEDED

        IWRITE  = IDBGE2 .NE. 6 .AND. IDBGE2 .LT. 1000
        IWRITE  = .NOT. IWRITE
        IF (KOUNT .EQ. 0) THEN
           KOUNT = 1
           IF (IWRITE)  THEN
              WRITE(JDEBUG,1000)
              WRITE(JDEBUG,1100)
              WRITE(JDEBUG,1200)
           ENDIF
        ENDIF

C       MARK ALL THE NODES FOR THE CASE FOR SUBSEQUENT UPDATING

        MARK(0) = 0
        DO 10 INODE = 1, NNODG2
           MARK(INODE) = 1
10      CONTINUE

C       LOOP OVER ALL THE CELLS AT THIS LEVEL AND UPDATE THE
C       CORRESPONDING NODES

        DO 100 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)

C           NODE/CELL ASSIGNMENTS

            ICELL = ICELTI (  JCELL)
            KSW   = ICELG2 (2,ICELL)
            KS    = ICELG2 (3,ICELL)
            KSE   = ICELG2 (4,ICELL)
            KE    = ICELG2 (5,ICELL)
            KNE   = ICELG2 (6,ICELL)
            KN    = ICELG2 (7,ICELL)
            KNW   = ICELG2 (8,ICELL)
            KW    = ICELG2 (9,ICELL)

C           CHECK IF UPDATING IS TO BE DONE AT THE SOUTHWESTERN NODE

            IF ( MARK(KSW) .NE. 0 ) THEN
                MARK(KSW) = 0
                DO 20 J  = 1, NEQNFL
                  DPENG2(J,KSW) = DPENG2(J,KSW) + CHNGE2(J,KSW)
C                 IF (IWRITE)  THEN
C                   IF ( ABS(CHNGE2(J,KSW)) .GT. 1. ) THEN
```

```
C                 WRITE(JDEBUG,1300) J, KSW, ITGL, CHNGE2(J,KSW)
C           -       ENDIF
C           -     ENDIF
                  CHNGE2(J,KSW) = 0.
20              CONTINUE
                CALL E2PRMT (KSW, 1)
              ENDIF

C             CHECK IF UPDATING IS TO BE DONE AT THE SOUTHERN EDGE

              IF ( MARK(KS) .NE. 0 ) THEN
                MARK(KS) = 0
                DO 30 J  = 1, NEQNFL
                  DPENG2(J,KS) = DPENG2(J,KS) + CHNGE2(J,KS)
                  CHNGE2(J,KS) = 0.
30              CONTINUE
                CALL E2PRMT (KS, 1)
              ENDIF

C             CHECK IF UPDATING IS TO BE DONE AT THE SOUTHEASTERN NODE

              IF ( MARK(KSE) .NE. 0 ) THEN
                MARK(KSE) = 0
                DO 40 J  = 1, NEQNFL
                  DPENG2(J,KSE) = DPENG2(J,KSE) + CHNGE2(J,KSE)
                  CHNGE2(J,KSE) = 0.
40              CONTINUE
                CALL E2PRMT (KSE, 1)
              ENDIF

C             CHECK IF UPDATING IS TO BE DONE AT THE EASTERN EDGE

              IF ( MARK(KE) .NE. 0 ) THEN
                MARK(KE) = 0
                DO 50 J  = 1, NEQNFL
                  DPENG2(J,KE) = DPENG2(J,KE) + CHNGE2(J,KE)
                  CHNGE2(J,KE) = 0.
50              CONTINUE
                CALL E2PRMT (KE, 1)
              ENDIF

C             CHECK IF UPDATING IS TO BE DONE AT THE NORTHEASTERN NODE

              IF ( MARK(KNE) .NE. 0 ) THEN
                MARK(KNE) = 0
                DO 60 J  = 1, NEQNFL
                  DPENG2(J,KNE) = DPENG2(J,KNE) + CHNGE2(J,KNE)
                  CHNGE2(J,KNE) = 0.
60              CONTINUE
                CALL E2PRMT (KNE, 1)
              ENDIF

C             CHECK IF UPDATING IS TO BE DONE AT THE NORTHERN EDGE

              IF ( MARK(KN) .NE. 0 ) THEN
                MARK(KN) = 0
                DO 70 J  = 1, NEQNFL
```

```
                DPENG2(J,KN) = DPENG2(J,KN) + CHNGE2(J,KN)
              -   CHNGE2(J,KN) = 0.
70            -  CONTINUE
                 CALL E2PRMT (KN, 1)
               ENDIF

C              CHECK IF UPDATING IS TO BE DONE AT THE NORTHWESTERN NODE

               IF ( MARK(KNW) .NE. 0 ) THEN
                 MARK(KNW) = 0
                 DO 80 J  = 1, NEQNFL
                   DPENG2(J,KNW) = DPENG2(J,KNW) + CHNGE2(J,KNW)
                   CHNGE2(J,KNW) = 0.
80               CONTINUE
                 CALL E2PRMT (KNW, 1)
               ENDIF

C              CHECK IF UPDATING IS TO BE DONE AT THE WESTERN EDGE

               IF ( MARK(KW) .NE. 0 ) THEN
                 MARK(KW) = 0
                 DO 90 J  = 1, NEQNFL
                   DPENG2(J,KW) = DPENG2(J,KW) + CHNGE2(J,KW)
                   CHNGE2(J,KW) = 0.
90               CONTINUE
                 CALL E2PRMT (KW, 1)
               ENDIF

100        CONTINUE
C
C          PRINT OUT PARAMETERS
C
           IF (IWRITE)  THEN
              ICELL  = ICELTI( ILVLTI(1,ITGL) )
              DTITGL = CELLTI(ICELL)
              WRITE(JDEBUG,1500) ITGL, DTITGL, DTMNTI
           ENDIF

C          -----------------
C          FORMAT STATEMENTS
C          -----------------

1000       FORMAT(//10X,'----------------------' )
1100       FORMAT(  10X,'DEBUG PRINT FROM E2UPDO' )
1200       FORMAT(  10X,'----------------------'/)
C1300      FORMAT(5X,'EQ. # ', I2, 5X, 'WEST NODE =', I5, 5X,
C        1           'TEMPORAL LEVEL =', I5, 5X, 'CHANGE =', G14.5)
C1400      FORMAT(5X,'EQ. # ', I2, 5X, 'EAST NODE =', I5, 5X,
C        1           'TEMPORAL LEVEL =', I5, 5X, 'CHANGE =', G14.5)
1500       FORMAT(5X,'ITGL=',I5,5X,'DT-ITGL =',G14.5,5X,'DTMNTI =',G14.5/)

           RETURN
           END
```

E2VARB

```
      SUBROUTINE E2VARB (TIME)

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'TVCOMN.INC'

C*********************************************************************
C
C       THE SUBROUTINE APPLIES THE PERIODIC BOUNDARY CONDITIONS.
C       AT FIRST ONLY OSCILLATIONS IN THE INLET MASS FLOW RATE
C       ARE CONSIDERED.  THE ROUTINE CAN BE GENERALIZED LATTER.
C
C*********************************************************************
C
C       COMPUTE THE TIME VARYING QUANTITIES
C
      OMEGAT = FREQTV*TIME*6.2831853
      SINWT  = SIN(OMEGAT)
      RHOU   = FLOWTV*(1.+AMPLTV*SINWT)
      INODE  = NODETV(1)
      DPENG2(2,INODE) = RHOU
      CALL E2PRMT(INODE,1)
      PRESPR = PRESG2(INODE)
      TEMPPR = TEMPG2(INODE)

      DO 1000 ITV = 2, NUMNTV
         INODE          = NODETV(ITV)
         DPENG2(2,INODE) = RHOU
         PRESG2(INODE)   = PRESPR
         TEMPG2(INODE)   = TEMPPR
1000  CONTINUE

      RETURN
      END
```

E2VECT

```
      SUBROUTINE E2VECT (ALVECT)

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'JACOMN.INC'
      INCLUDE 'PRCOMN.INC'

      DIMENSION UTOP(MEQNFL), TOTAL(MEQNFL), F2JACO(MEQNFL),
     1          UBOT(MEQNFL), DUMY (MEQNFL), ALVECT(MEQNFL,MEQNFL)
```

```
C***********************************************************************
C
C       THIS SUBROUTINE COMPUTES THE LEFT EINVECTOR MATRIX OF THE FLUX
C       F IN THE STREAMWISE COORDINATE SYSTEM AT A GIVEN PLACE.

C***********************************************************************
C
C       COMPUTE THE CENTRAL DIFFERNCE NUMERICAL JACOBIANS BUT FIRST
C       SET UP THE QUANTITIES NEEDED TO ACCOMPLISH THIS

        DO 10 IQ = NEQBAS+1, NEQNFL
           DELTA     = 0.01*DPENJA(IQ)
           IF (DELTA .EQ. 0.) DELTA = 0.01
           UTOP(IQ)  = DPENJA(IQ) + DELTA
           UBOT(IQ)  = DPENJA(IQ) - DELTA
           TOTAL(IQ) = 2.*DELTA
10      CONTINUE
C
        DO 20 IQ = NEQBAS+1, NEQNFL

C          COMPUTE VALUES AT TOP

           UDUMMY     = DPENJA(IQ)
           DPENJA(IQ) = UTOP(IQ)
           CALL         FLBGF2
           F2TOP      = BGF2JA

C          COMPUTE VALUES AT BOTTOM

           DPENJA(IQ) = UBOT(IQ)
           CALL         FLBGF2
           F2BOT      = BGF2JA

C          RESET THE VALUE OF THE DEPENDENT VARIABLE

           DPENJA(IQ) = UDUMMY

C          NOW TAKE CENTRAL DIFFERENCE

           F2JACO(IQ) = (F2TOP - F2BOT)/TOTAL(IQ)

20      CONTINUE
C
C       NOW COMPUTE THE ANALYTIC JACOBIANS; INITIALIZE THE VALUES
C       UCOMPR, VCOMPR, GAMAPR, YSPEPR ETC.
C
        CALL FLBGF2

        U2     = UCOMPR*UCOMPR
        V2     = VCOMPR*VCOMPR
        SONDPR = SQRT (GAMAPR*PRESPR/RHORPR)
        GM1    = GAMAPR - 1.
        GM3    = GM1 - 2.
        PAEBR  = (BEPSPR+PRESPR)/RHORPR
C
        F2JACO(1) = 0.5*(GM3*U2 + GM1*V2)
        F2JACO(2) = -GM3*UCOMPR
```

```
      F2JACO(3) = -GM1*VCOMPR
      F2JACO(4) =  GM1
C          -
C     F4JAC1 = UCOMPR*(F2JACO(1) + U2 - PAEBR)
      F4JAC2 = PAEBR - 2.*U2 + UCOMPR*F2JACO(2)
C     F4JAC3 = UCOMPR*F2JACO(3)
C     F4JAC4 = UCOMPR*(F2JACO(4) + 1.)
C
C     INITIALIZE THE EIGENVECTOR MATRIX
C
      DO 40 IQ = 1, NEQNFL
        DO 30 JQ = 1, NEQNFL
          ALVECT(IQ,JQ) = 0.
30      CONTINUE
40    CONTINUE

C     EIGENVECTERS FOR U - A

      ALVECT(1,2) = -(SONDPR/F2JACO(4) + UCOMPR)
      ALVECT(1,3) = -VCOMPR
      ALVECT(1,4) = 1.
      SUMALY      = 0.

      DO 50 JQ = NEQBAS+1, NEQNFL
        IQ           = JQ - NEQBAS
        ALVECT(1,JQ) = F2JACO(JQ)/F2JACO(4)
        ALVECT(2,JQ) = ALVECT(1,JQ)
        SUMALY       = SUMALY + ALVECT(1,JQ)*YSPEPR(IQ)
50    CONTINUE

      ALVECT(1,1) = ALVECT(1,2)*(UCOMPR - SONDPR - F2JACO(2)) +
     1              V2 - F4JAC2 - SUMALY

C     EIGENVECTERS FOR U + A

      ALVECT(2,2) = (SONDPR/F2JACO(4) - UCOMPR)
      ALVECT(2,3) = -VCOMPR
      ALVECT(2,4) = 1.
      ALVECT(2,1) = ALVECT(2,2)*(UCOMPR + SONDPR - F2JACO(2)) +
     1              V2 - F4JAC2 - SUMALY

C     EIGENVECTERS FOR U FOR Y-MOMENTUM EQUATION

      ALVECT(3,1) = -VCOMPR
      ALVECT(3,3) = 1.

C     EIGENVECTERS FOR U FOR CONTINUITY EQUATION

      ALVECT(4,1) = UCOMPR*(F2JACO(2) - UCOMPR) - F4JAC2
      ALVECT(4,2) = -UCOMPR
      ALVECT(4,4) = 1.

C     EIGENVECTERS FOR U FOR SPECIES EQUATION

      DO 60 IQ = NEQBAS+1, NEQNFL
        ALVECT(IQ,1 ) = -YSPEPR(IQ-NEQBAS)
        ALVECT(IQ,IQ) = 1.
```

```
60      CONTINUE
C       GRAM-SCHMIDT PROCESS
C       CALL GRAMSM (ALVECT,DUMY,NEQNFL,MEQNFL)
        RETURN
        END
```

E2ZERO

```
        SUBROUTINE E2ZERO
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'E2COMN.INC'
C***********************************************************************
C       THIS SUBROUTINE INITIALIZES THE CHANGES AT ALL THE NODES TO
C       ZEROS, IT IS DIFFERENT FROM E2ZER1 IN THE SENSE THAT IT IS
C       GLOBAL.
C***********************************************************************
        DO 10 J = 1, NEQNFL
          DO 10 IN = 1, NNODG2
C           IF (CHNGE2(J,IN) .NE. 0. ) THEN
              CHNGE2(J,IN) = 0.
C           ENDIF
10      CONTINUE
        RETURN
        END
```

ERINIT

```
        SUBROUTINE ERINIT
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'CHCOMN.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'H2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'TICOMN.INC'
```

```
C***********************************************************************
C
C       THIS SUBROUTINE PERFORMS THE INITIAL ERROR CHECKS FOR THE TWO-
C       DIMENSIONAL CASE.
C
C***********************************************************************
C
C       CHECK MAXIMUM NO. OF EQUATIONS, I.E., IF NEQNFL < MEQNFL ?
C
        IF (NEQNFL .LE. 0 .OR. NEQNFL .GT. MEQNFL) THEN
           ZER1 = NEQNFL
           ZER2 = MEQNFL
           CALL ERRORM (24,'ERINIT','NEQNFL',ZER1,'MEQNFL',ZER2,JPRINT,
     1             'NUMBER OF EQUATIONS IS SET WRONG')
        ENDIF
C
C       MAXIMUM GIVEN SPATIAL LEVEL OF CELLS
C
        IF (MALVG2 .LT. 0 .OR. MALVG2 .GT. MLVLG2) THEN
           ZER1 = MALVG2
           ZER2 = MLVLG2
           CALL ERRORM (25,'ERINIT','MALVG2',ZER1,'MLVLG2',ZER2,JPRINT,
     1             'ERROR IN MAXIMUM GIVEN SPATIAL LEVEL')
        ENDIF
C
C       CHEMISTRY TYPE SELECTOR KROGER
C
        IF (KROGER .LT. 0 .OR. KROGER .GT. 3) THEN
           ZER1 = KROGER
           ZER2 = 3
           CALL ERRORM (26,'ERINIT','KROGER',ZER1,'MODMAX',ZER2,JPRINT,
     1             'CHEMISTRY MODEL IS SET WRONG')
        ENDIF
C
C       MAXIMUM NUMBER OF CELLS TO BE EXTENDED
C
        IF (NXTDA2 .LT. 0 .OR. NXTDA2 .GT. 7) THEN
           ZER1 = NXTDA2
           ZER2 = 7
           CALL ERRORM (27,'ERINIT','NXTDA2',ZER1,'MAXEXT',ZER2,JPRINT,
     1             'MAXIMUM NUMBER OF EXTENSION CELLS IS WRONG')
        ENDIF
C
C       METHOD OF SPATIAL ADAPTATION
C       METHA2 IS SET ZERO IF YOU WANT TO SKIP ADAPTATION LOOP
C
        IF (METHA2 .LT. 0 .OR. METHA2 .GT. 6) THEN
           ZER1 = METHA2
           ZER2 = MTYPA2
           CALL ERRORM (28,'ERINIT','METHA2',ZER1,'MTYPA2',ZER2,JPRINT,
     1             'METHOD OF ADAPTATION IS SET WRONG')
        ENDIF
C
C       CHECK THE SPATIAL ADAPTATION CRITERIA VARIABLES
C       K1ADA2 = 0, IS NOT NEEDED SINCE THE SPATIAL ADAPTATION IS
C       BY-PASSED BY SETTING MITRA2 = 0
C
```

```
      IF (K1ADA2 .LE. 0 .OR. K1ADA2 .GT. NEQNFL) THEN
         ZER1 = K1ADA2
         ZER2 = NEQNFL
         CALL ERRORM (29,'ERINIT','K1ADA2',ZER1,'NEQNFL',ZER2,JPRINT,
     1            'SPATIAL ADAPTATION CRITERIA IS NOT SET CORRECTLY')
      ENDIF
C
      IF (K2ADA2 .LT. 0 .OR. K2ADA2 .GT. 100+NEQNFL) THEN
         ZER1 = K2ADA2
         ZER2 = NEQNFL
         CALL ERRORM (29,'ERINIT','K2ADA2',ZER1,'NEQNFL',ZER2,JPRINT,
     1            'SPATIAL ADAPTATION CRITERIA IS NOT SET CORRECTLY')
      ENDIF
C
C     ERROR CHECK FOR G2CLPO AND G2DIVO
C
      IF (KCHKA2 .LT. 0 .OR. KCHKA2 .GT. 15) THEN
         ZER1 = KCHKA2
         ZER2 = 15.
         CALL ERRORM (30,'ERINIT','KCHKA2',ZER1,'MAXVAL',ZER2,JPRINT,
     1            'SUPERCELL/NEIGHBOUR-CELL CHECK INDICATOR IS WRONG')
      ENDIF
C
C     SPATIAL ADAPTATION TUNING PARAMETERS
C
      IF (BETAA2 .LE. 0. .OR. BETAA2 .GT. 0.5) THEN
         ZER1 = BETAA2
         ZER2 = ALPHA2
         CALL ERRORM (31,'ERINIT','BETAA2',ZER1,'ALPHA2',ZER2,JPRINT,
     1            'SPATIAL ADAPTATION PARAMETER IS SET WRONG')
      ENDIF
C
      IF (GAMMA2 .LE. 0. .OR. GAMMA2 .GT. 0.9) THEN
         ZER1 = GAMMA2
         ZER2 = DELTA2
         CALL ERRORM (31,'ERINIT','GAMMA2',ZER1,'DELTA2',ZER2,JPRINT,
     1            'SPATIAL ADAPTATION PARAMETER IS SET WRONG')
      ENDIF
C
C     TYPE OF CONVERGENCE HISTORY
C     KONVE2 IS SET ZERO FOR TIME ACCURATE PROBLEMS (SKIP CONVERGENCE)
C
      IF (KONVE2 .LT. 0 .OR. KONVE2 .GT. 3) THEN
         ZER1 = KONVE2
         ZER2 = 3.
         CALL ERRORM (32,'ERINIT','KONVE2',ZER1,'MAXVAL',ZER2,JPRINT,
     1            'CONVERGENCE TYPE IS UNKNOWN')
      ENDIF
C
C     EQUATION USED IN THE ABOVE CONVERGENCE TYPE
C
      IF (KEQNE2 .LE. 0 .OR. KEQNE2 .GT. NEQNFL) THEN
         ZER1 = KEQNE2
         ZER2 = NEQNFL
         CALL ERRORM (33,'ERINIT','KEQNE2',ZER1,'NEQNFL',ZER2,JPRINT,
     1            'CONVERGENCE EQUATION SELECTOR IS WRONG')
      ENDIF
```

```
C
C       COURANT NUMBER
C
        IF (CFLNTI .LE. 0. .OR. CFLNTI .GT. 1.) THEN
           ZER1 = CFLNTI
           ZER2 = 1.
           CALL ERRORM (34,'ERINIT','CFLNTI',ZER1,'MAXVAL',ZER2,JPRINT,
     1             'CFL NUMBER IS SET WRONG')
        ENDIF
C
C       IMPLICIT/EXPLICIT SCHEME SELECTOR
C
        IF (IMPLTI .LT. 0 .OR. IMPLTI .GT. 2) THEN
           ZER1 = IMPLTI
           ZER2 = 0.
           CALL ERRORM (35,'ERINIT','IMPLTI',ZER1,'NIL   ',ZER2,JPRINT,
     1             'IMPLICIT/EXPLICIT SCHEME SELECTOR IS WRONG')
        ENDIF
C
C       CHECK THE TEMPORAL RESOLUTION CRITERIA VARIABLE
C       KADPTI IS SET ZERO IF YOU WANT TO SKIP IT
C
        IF (KADPTI .LT. 0 .OR. KADPTI .GT. NEQNFL) THEN
           IF (KADPTI .EQ. 99) GOTO 10
           ZER1 = KADPTI
           ZER2 = NEQNFL
           CALL ERRORM (36,'ERINIT','KADPTI',ZER1,'NEQNFL',ZER2,JPRINT,
     1             'TEMPORAL ADAPTATION CRITERIA IS NOT SET CORRECTLY')
        ENDIF
C
C       MAXIMUM GIVEN TEMPORAL LEVEL OF CELLS
C
10        IF (NGIVTI .LT. 0 .OR. NGIVTI .GT. MMAXTI) THEN
           ZER1 = NGIVTI
           ZER2 = MMAXTI
           CALL ERRORM (37,'ERINIT','NGIVTI',ZER1,'MMAXTI',ZER2,JPRINT,
     1             'ERROR IN MAXIMUM GIVEN TEMPORAL LEVEL')
        ENDIF
C
C       CHECK THE MASS FRACTION OF THE INERT SPECIES
C
        IF (YNRTCH .LT. 0. .OR. YNRTCH .GT. 1.) THEN
           ZER1 = YNRTCH
           ZER2 = 1.
           CALL ERRORM (38,'ERINIT','YNRTCH',ZER1,'MAXVAL',ZER2,JPRINT,
     1             'INERT SPECIES MASS FRACTION ERROR')
        ENDIF
C
C       MAXIMUM NUMBER OF SPECIES EQUATIONS
C
        IF (NEQSCH .LT. 0 .OR. NEQSCH .GT. MEQNFL-NEQBAS) THEN
           ZER1 = NEQSCH
           ZER2 = MEQNFL - NEQBAS
           CALL ERRORM (39,'ERINIT','NEQSCH',ZER1,'MAXVAL',ZER2,JPRINT,
     1             'NUMBER OF SPECIES EQUATIONS IN ERROR')
        ENDIF
C
```

```
C       RESTART PARAMETER; 0:FRESH  1:RESTART  2:C2HELP
C
        IF (KSRTE2 .LT. 0 .OR. KSRTE2 .GT. 1003) THEN
           ZER1 = KSRTE2
           ZER2 = 2.
           CALL ERRORM (40,'ERINIT','KSRTE2',ZER1,'MAXVAL',ZER2,JPRINT,
     1             'ERROR IN RESTART PARAMETER')
        ENDIF
C
C       NUMBER OF INERT SPECIES
C
        IF (NINRCH .LT. 0 .OR. NINRCH .GT. NSPECH) THEN
           ZER1 = NINRCH
           ZER2 = NSPECH
           CALL ERRORM (41,'ERINIT','NINRCH',ZER1,'NSPECH',ZER2,JPRINT,
     1             'NUMBER OF INERT SPECIES IS WRONG')
        ENDIF
C
C       CHECK IF FUEL INJECTION IS NEEDED FOR ANYTHING OTHER THAN THE
C       ROGERS AND CHINITZ MODEL
C
        IF (KROGER .NE. 1 .AND. IADDH2 .NE. 0) THEN
           ZER1 = KROGER
           ZER2 = IADDH2
           CALL ERRORM (42,'ERINIT','KROGER',ZER1,'IADDH2',ZER2,JPRINT,
     1      'FUEL INJECTION IS ONLY ALLOWED FOR ROGERS & CHINITZ MODEL')
        ENDIF

        RETURN
        END
```

FLBGF2

```
        SUBROUTINE FLBGF2

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'CHCOMN.INC'
        INCLUDE 'FLCOMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'JACOMN.INC'
        INCLUDE 'PRCOMN.INC'

C***********************************************************************

C       THIS SUBROUTINE COMPUTES THE SECOND COMPONENT OF THE FLUX VECTOR
C       F IN THE USUAL (X,Y) COORDINATES.  THIS IS NEEDED HERE FOR
C       COMPUTING JACOBIANS OF F2.  WE DO NOT USE E2SOUR BECAUSE THAT
C       ROUTINE ALSO SOLVES FOR THE SOURCE TERMS WHICH ARE NOT NEEDED
C       HERE.  THE TEMPORALLY VARYING VARIABLES ARE SUPPOSED TO BE
C       STORED IN THE JA COMMON VARIABLES, I.E., IN DPENJA(J).

C***********************************************************************
```

```
      RHORPR = DPENJA(1)
      UCOMPR = DPENJA(2)/DPENJA(1)
      VCOMPR = DPENJA(3)/DPENJA(1)
      BEPSPR = DPENJA(4)
      BEU    = BEPSPR/RHORPR
      VEL02U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BEU
      VEL02  = FMREFL*VEL02U

C     COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

      SUMY = 0.

      DO 10 IS = 1, NEQSCH
          JS         = NEQBAS + IS
          YSPEPR(IS) = DPENJA(JS)/DPENJA(1)
          SUMY       = SUMY + YSPEPR(IS)
10    CONTINUE

      YNEXT            = 1. - SUMY - YNRTCH
      IF (YNEXT .LT. 0.) YNEXT = 0.
      YSPEPR(NEQSCH+1) = YNEXT

      SYSHFS = 0.
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
C     COMPUTE THE TEMPERATURE IN DEGREE K AND ALSO
C
      DO 20 IS = 1, NSPECH
         SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
         SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
         SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
         BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
20    CONTINUE
C
C     COMPUTE TEMPERATURE IN DEGREE K AND NORMALIZE IT
C
      BIGBM   = SYSCPS - UGASFL*SYSBMS
      BIGCM   = BE - 0.5*VEL02 - SYSHFS + TREFCH*SYSCPS
     1             + 0.5*TREFCH*TREFCH*BIGAM
      IF (BIGAM .LT. 1.E-10) THEN
         TEMP   = BIGCM/BIGBM
      ELSE
         DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
         TEMP   = ( SQRT(DISCRI)-BIGBM )/BIGAM
      ENDIF

      BIGAMT = BIGAM *TEMP
      SYSCVS = BIGBM + BIGAMT
      TEMPU = TEMP/TREFFL
C
```

```
C       COMPUTE THE DIMENSIONLESS PRESSURE
C
        PRESPR = RHORPR*TEMPU*AMWTFL*SYSBMS

C       COMPUTE THE FLUX VARIABLE

        BGF2JA = DPENJA(2)*UCOMPR + PRESS
        GAMAPR = (SYSCPS+BIGAMT)/SYSCVS
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGFL .NE. 5 .AND. IDBGFL .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) BGF2JA

        DO 30 IS = 1, NEQNFL
          WRITE(JDEBUG,1400) IS, DPENJA(IS)
30      CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'-----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM FLBGF2' )
1200    FORMAT(  10X,'-----------------------'/)
1300    FORMAT(  5X, 'BGF2JA=', G14.5)
1400    FORMAT(  5X,I5,G14.5)

        RETURN
        END
```

FLINI2

```
        SUBROUTINE FLINI2

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'CHCOMN.INC'
        INCLUDE 'FLCOMN.INC'
        INCLUDE 'HEXCOD.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'KYCOMN.INC'

C***********************************************************************

C       THIS SUBROUTINE INITIALIZES THE COMMON BLOCK FLCOMN IT ALSO
C       INITIALIZES THE DEPENDENT VARIABLES OVER ALL THE NODES TO A
C       UNIFORM FLOW
```

```
C***********************************************************************
C       GET THE VALUES SET BY GETKY2 SUBROUTINE OR THE DEFAULT VALUES

        AMCHFL    = APASKY( 5)
        RHORFL    = APASKY( 6)
        TREFFL    = APASKY( 7)
        PRESFL    = APASKY( 9)
        DISTFL    = APASKY(11)

        IDBGFL    = IPASKY(13)
        NEQBAS    = 4

C       SET UP THE NUMBER OF EQUATIONS TO BE SOLVED
        NEQNFL = NSPECH + NEQBAS  - NINRCH - 1

C       SETUP THE NUMBER OF SPECIES EQUATIONS
        NEQSCH = NEQNFL - NEQBAS

C       SET UP THE UNIVERSAL GAS CONSTANT
        UGASFL = 8.31434E03
C
C       COMPUTE THE MOLECULAR MASS AND GAMMA OF THE MIXTURE
C
        SYSCPS = 0.
        SYSBMS = 0.
        BIGAM  = 0.
C
        DO 10 IS = 1, NSPECH
            SYSCPS = SYSCPS + YSPECH(IS)*SPCPCH(IS)
            SYSBMS = SYSBMS + YSPECH(IS)/AMWTCH(IS)
            BIGAM  = BIGAM  + YSPECH(IS)*SPBSCH(IS)
10      CONTINUE

C       COMPUTE THE OTHER REFERENCE QUANTITIES
C       AT-LEAST TWO OF THE FOLLOWING REFERENCE MUST BE SET BY GETKY2
C                   TREFFL,  PRESFL,  RHORFL

        IF (PRESFL .NE. 1.) KPRT = IOR(KL0001,KPRT)
        IF (RHORFL .NE. 1.) KPRT = IOR(KL0002,KPRT)
        IF (TREFFL .NE. 1.) KPRT = IOR(KL0004,KPRT)
        IF (KPRT   .EQ. 7 ) KPRT = 3

        UGASCO = UGASFL*SYSBMS

        IF (KPRT .EQ. 3) TREFFL = PRESFL/(UGASCO*RHORFL)
        IF (KPRT .EQ. 5) RHORFL = PRESFL/(UGASCO*TREFFL)
        IF (KPRT .EQ. 6) PRESFL = RHORFL*(UGASCO*TREFFL)

        UREFFL = SQRT(PRESFL/RHORFL)
        FMREFL = UREFFL**2
        WDREFL = RHORFL*UREFFL/DISTFL

        AMWTFL = 1./SYSBMS
        SYSCPS = SYSCPS + BIGAM*TREFFL
        SYSCVS = SYSCPS - UGASFL*SYSBMS
        GAMAFL = SYSCPS/SYSCVS
```

```
C       COMPUTE THE MASS FRACTIONS YNRTCH OF THE INERT SPECIES, WHICH
C       ARE SUPPOSED TO BE STORED AT THE TRAILING END OF YSPECH.

        YNRTCH = 0.

        NFINAL = NSPECH - NINRCH + 1

        DO 20 IS = NSPECH, NFINAL, -1
           YNRTCH = YNRTCH + YSPECH(IS)
20      CONTINUE

        WRITE(JOUTAL,1300)
        WRITE(JOUTAL,1400) PRESFL, TREFFL, RHORFL, UGASFL,
     1                     AMCHFL, DISTFL, UREFFL, FMREFL
        WRITE(JOUTAL,1450) WDREFL, AMWTFL, YNRTCH,
     1                     GAMAFL, NEQNFL, NEQSCH, NEQBAS, NINRCH

        WRITE(JOUTAL,1500)
        WRITE(JOUTAL,1700)

        DO 30 IS = 1,NSPECH
          WRITE(JOUTAL,1800) IS,YSPECH(IS)
30      CONTINUE

C
C       DEBUG PRINT
C
        IF (IDBGFL .NE. 3 .AND. IDBGFL .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1400) PRESFL, TREFFL, RHORFL, UGASFL,
     1                     AMCHFL, DISTFL, UREFFL, FMREFL
        WRITE(JDEBUG,1450) WDREFL, AMWTFL, YNRTCH,
     1                     GAMAFL, NEQNFL, NEQSCH, NEQBAS, NINRCH
        WRITE(JDEBUG,1600) UGASCO, NEQSCH, NEQBAS
        WRITE(JDEBUG,1700)

        DO 40 IS = 1,NSPECH
          WRITE(JDEBUG,1800) IS,YSPECH(IS)
40      CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM FLINI2' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(' THE DIMENSIONAL QUANTITIES WHICH PRODUCE ',
     1   'THIS OUTPUT ARE'//)
1400    FORMAT( 5X, 'PRESFL = ', G14.5, 3X, 'PA            ',
     1          5X, 'TREFFL = ', G14.5, 3X, 'K             '/
     2          5X, 'RHORFL = ', G14.5, 3X, 'KG/(M.M.M)    ',
     3          5X, 'UGASFL = ', G14.5, 3X, 'J/KMOL/K      '/
```

```
     4          5X, 'AMCHFL = ', G14.5,     17X             ,
     5      -   5X, 'DISTFL = ', G14.5, 3X, 'M             '/
     6      -   5X, 'UREFFL = ', G14.5, 3X, 'M/S           ',
     7          5X, 'FMREFL = ', G14.5, 3X, '(M.M)/(S.S)   '   )
1450    FORMAT( 5X, 'WDREFL = ', G14.5, 3X, 'KG/(M3.S)     ',
     2          5X, 'AMWTFL = ', G14.5, 3X, 'KG/KMOLE      '/
     3          5X, 'YNRTCH = ', G14.5, 3X, '              ',
     4          5X, 'GAMAFL = ', G14.5, 3X, '              '/
     5          5X, 'NEQNFL = ', I5,   31X, 'NEQSCH = ', I5,/
     6          5X, 'NEQBAS = ', I5,   31X, 'NINRCH = ', I5,// )
1500    FORMAT(' THE REST OF THE OUTPUT IS IN NON-DIMENSIONAL FORM'//)
1600    FORMAT(5X,'UGASCO = ',G14.5,3X,'J/KG/K        '/
     1        5X,'NUMBER OF SPECIES EQUATIONS =',I5,10X,
     2           'NUMBER OF BASIC CONSERVATION  EQUATIONS =',I5 )
1700    FORMAT(/5X,'REFERENCE SPECIES MASS FRACTIONS'/
     1        10X,'SPECIES',3X,'MASS FRACTION')
1800    FORMAT(  5X,I5,G14.5)

        RETURN
        END
```

FRINIT

```
        SUBROUTINE FRINIT

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'CHCOMN.INC'
        INCLUDE 'FLCOMN.INC'
        INCLUDE 'FRCOMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'KYCOMN.INC'
        INCLUDE 'PRCOMN.INC'

C***********************************************************************

C       THIS SUBROUTINE INITIALIZES THE COMMON BLOCK FRCOMN, WHICH HOLDS
C       FREE STREAM CONDITIONS.  COMMON BLOCK FLCOMN HOLDS CORRESPONDING
C       DIMENSIONAL VALUES.

C***********************************************************************
C
C       GET THE VALUES SET BY GETKY2 SUBROUTINE OR THE DEFAULT VALUES
C
C       FREE STREAM DENSITY
        RHORFR = APASKY(30)
C
C       FREE STREAM VELOCITY COMPONENTS
        UCOMFR = APASKY(31)
        VCOMFR = APASKY(32)
C
C       FREE STREAM PRESSURE
        PRESFR = APASKY(33)
```

```
C
C       BACK-PRESSURE RATIO
        PBPIFR = APASKY(21)
C
C       SET THE DEBUG PARAMETER FOR FR ROUTINES
        IDBGFR = IPASKY(21)

C       WANT TO USE PERIDIC BOUNDARY CONDITIONS
        KPERFR = IPASKY(35)
C
C       MAXIMUM NUMBER OF CYCLES
        MCYCFR = IPASKY(36)
C
C       CURRENT NUMBER OF CYCLES
        NCYCFR = 0
C
C       ------------------
C       FREE STREAM VECTOR
C       ------------------
C
C       SAVE THE MASS FRACTIONS
C
        DO 10 IS = 1, NSPECH
           YSPEPR(IS) = DPENFR(IS)
10      CONTINUE
C
C       DETERMINE THE FIRST COMPONENT OF THE FREE STREAM
C       DEPENDENT VARIABLE VECTOR
C
        DPENFR(1) = RHORFR
C
C       COMPUTE THE COMPONENTS PERTAINING TO SPECIES EQUATIONS
C
        DO 20 IS = 1, NEQSCH
           JS         = NEQBAS + IS
           DPENFR(JS) = RHORFR*YSPEPR(IS)
20      CONTINUE
C
C       COMPUTE MIXTURE SPECIFIC HEATS
C
        SYSHFS = 0.
        SYSCPS = 0.
        SYSBMS = 0.
        BIGAM  = 0.

        DO 30 IS = 1, NSPECH
            SYSHFS = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
            SYSCPS = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
            SYSBMS = SYSBMS + YSPEPR(IS)/AMWTCH(IS)
            BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
30      CONTINUE
C
C       COMPUTE THE DIMENSIONAL TEMPERATURE
C
        UGASCO = UGASFL*SYSBMS
        TREFFR = (PRESFR*PRESFL)/(UGASCO*RHORFR*RHORFL)
C
```

```
C       SEE IF YOU WANT TO COMPUTE THE VELOCITY COMPONENTS FROM
C       THE GIVEN MACH NUMBER
C          -
        IF (UCOMFR .EQ. 0.) THEN
           SYSCVS = SYSCPS + BIGAM*TREFFR - UGASFL*SYSBMS
           GAMAPR = (SYSCPS+BIGAM*TREFFR)/SYSCVS
           SONDPR = GAMAPR*PRESFR/RHORFR
           UCOMFR = AMCHFL*SQRT(SONDPR)
        ENDIF
C
        VEL02I    = UCOMFR*UCOMFR + VCOMFR*VCOMFR
        BEPSPR    = SYSHFS + (TREFFR-TREFCH)*SYSCPS
     1                      - UGASFL*TREFFR*SYSBMS
     1                      + 0.5*(TREFFR*TREFFR-TREFCH*TREFCH)*BIGAM
        BEPSPR    = BEPSPR/FMREFL + 0.5*VEL02I
        DPENFR(2) = RHORFR*UCOMFR
        DPENFR(3) = RHORFR*VCOMFR
        DPENFR(4) = RHORFR*BEPSPR
C
C       -----------
C       DEBUG PRINT
C       -----------
C
        IF (IDBGFR .NE. 1 .AND. IDBGFR .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) RHORFR, UCOMFR, VCOMFR, PRESFR, PBPIFR,
     1                     GAMAPR, SONDPR, BEPSPR, TREFCH, TREFFR,
     2                     IDBGFR, KPERFR, MCYCFR
        WRITE(JDEBUG,1400)

        DO 40 IS = 1,NSPECH
          WRITE(JDEBUG,1500) IS,YSPEPR(IS)
40      CONTINUE

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM FRINIT' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT( 5X, 'RHORFR = ', G14.5, 10X, 'UCOMFR = ', G14.5/
     1          5X, 'VCOMFR = ', G14.5, 10X, 'PRESFR = ', G14.5/
     2          5X, 'PBPIFR = ', G14.5, 10X, 'GAMAPR = ', G14.5/
     3          5X, 'SONDPR = ', G14.5, 10X, 'BEPSPR = ', G14.5/
     4          5X, 'TREFFR = ', G14.5, 10X, 'TREFCH = ', G14.5/
     5          5X, 'IDBGFR = ', I5,     5X, 'KPERFR = ', I5   ,
     6          5X, 'MCYCFR = ', I5                             )
1400    FORMAT(/5X,'FREE STREAM SPECIES MASS FRACTIONS'/
     1         10X,'SPECIES',3X,'MASS FRACTION')
1500    FORMAT(  5X,I5,G14.5)

        RETURN
        END
```

FRSOUR

```
      SUBROUTINE FRSOUR

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'JACOMN.INC'
      INCLUDE 'PRCOMN.INC'

      DIMENSION WREACT(MREACH)
      DOUBLE PRECISION PROD1, PROD2, RHOD, CONCEN(MSPECH)

C*********************************************************************

C     THIS FUNCTION COMPUTES THE SOURCE TERMS AT A GIVEN LOCATION.

C*********************************************************************

      RHORPR = DPENJA(1)
      UCOMPR = DPENJA(2)/DPENJA(1)
      VCOMPR = DPENJA(3)/DPENJA(1)
      BEPSPR = DPENJA(4)
      BEU    = BEPSPR/RHORPR
      VEL02U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BEU
      VEL02  = FMREFL*VEL02U
      RHOD   = RHORPR*RHORFL

C     COMPUTE THE MASS FRACTIONS FOR EACH SPECIES

      SUMY = 0.

      DO 10 IS = 1, NEQSCH
          JS         = NEQBAS + IS
          YSPEPR(IS) = DPENJA(JS)/DPENJA(1)
          SUMY       = SUMY + YSPEPR(IS)
10    CONTINUE

      YNEXT            = 1. - SUMY - YNRTCH
      IF (YNEXT .LT. 0.) YNEXT = 0.
      YSPEPR(NEQSCH+1) = YNEXT
C
      SYSHFS = 0.
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
```

```
C       COMPUTE THE TEMPERATURE IN DEGREE K AND ALSO
C       COMPUTE THE CONCENTRATIONS OF ALL THE SPECIES IN KMOL/(M**3)
C
        DO 20 IS = 1, NSPECH
           SYSHFS     = SYSHFS + YSPEPR(IS)*FMHTCH(IS)
           SYSCPS     = SYSCPS + YSPEPR(IS)*SPCPCH(IS)
           SYSBMS     = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
           BIGAM      = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
           CONCEN(IS) = RHOD*YSPEPR(IS)*RAMWCH(IS)
           BIGWJA(IS) = 0.
20      CONTINUE
C
C       COMPUTE TEMPERATURE IN DEGREE K AND SOME RELATED QUANTITIES

        BIGBM   = SYSCPS - UGASFL*SYSBMS
        BIGCM   = BE - 0.5*VEL02 - SYSHFS + TREFCH*SYSCPS
     1                      + 0.5*TREFCH*TREFCH*BIGAM
        IF (BIGAM .LT. 1.E-10) THEN
           TEMPD  = BIGCM/BIGBM
        ELSE
           DISCRI = BIGBM*BIGBM + 2.*BIGAM*BIGCM
           TEMPD  = ( SQRT(DISCRI)-BIGBM )/BIGAM
        ENDIF

        BIGAMT = BIGAM *TEMPD
        SYSCVS = BIGBM + BIGAMT
        GAMAPR = (SYSCPS+BIGAMT)/SYSCVS
        ALOGT  = LOG(ABS(TEMPD))
        RTEMP  = 1./TEMPD
C
C       NORMALIZE THE TEMPERATURE
C
        TEMPU = TEMPD/TREFFL
C
C       COMPUTE THE DIMENSIONLESS PRESSURE
C
        PRESPR = RHORPR*TEMPU*AMWTFL*SYSBMS

C       BY-PASS THE REACTION CALCULATIONS IF TEMPERATURE IS LESS THAN
C       TRIGGER TEMPERATURE

        IF (TEMPD .LT. TRIGCH) RETURN
        RECWDR = 1./WDREFL
C
C       CORRECT THE RATE COEFFICIENTS FOR ROGERS AND CHINITZ MODEL
C
        IF (KROGER .EQ. 1) THEN
           IF (YSPEPR(3) .LE. 0.) RETURN
           PHI       =  YSPEPR(3)*34.048/(1.-YSPEPR(3))
           IF (PHI .LT. 0.1 ) PHI = 0.1
           IF (PHI .GT. 2.0 ) PHI = 2.0
           RPHI      =  1./PHI
           TENLOG    =  LOG(10.)
           A1PHI     =  8.917*PHI + 31.433*RPHI - 28.95
           A2PHI     = -0.833*PHI +  1.333*RPHI +  2.00
           PREFCH(1) =  LOG(A1PHI) + 44.*TENLOG
           PREFCH(2) =  LOG(A2PHI) + 58.*TENLOG
```

```
      PREBCH(1) =  PREFCH(1) - PREECH(1)
      PREBCH(2) =  PREFCH(2) - PREECH(2)
C
C     USE THE FIRST REACTION AS EQUILIBRIUM REACTION, IF THE
C     CONCENTRATIONS ARE FAR AWAY FROM EQUILIBRIUM

      AKEQ   = 117.31948*EXP(-8992./TEMPD)
      YOHEQ  = YSPEPR(3)*YSPEPR(1)*AKEQ
      YOHEQ  = SQRT(YOHEQ)
      DELTAY = YOHEQ-YSPEPR(2)

      IF (ABS(DELTAY) .GT. 0.01*YMAXCH(2) .AND.
     1      YSPEPR(4) .LT. 0.50*YMAXCH(4)) THEN
          DELTAY    = 0.5*DELTAY*RAMWCH(2)
          YO2EQ     = YSPEPR(1) - AMWTCH(1)*DELTAY
          YH2EQ     = YSPEPR(3) - AMWTCH(3)*DELTAY
          CONCEN(1) = RHOD*YO2EQ*RAMWCH(1)
          CONCEN(2) = RHOD*YOHEQ*RAMWCH(2)
          CONCEN(3) = RHOD*YH2EQ*RAMWCH(3)
      ENDIF
C
C     REACTION # 1
C
      ALNKFR    = PREFCH(1) + EXPFCH(1)*ALOGT - ENEFCH(1)*RTEMP
      ALNKBR    = PREBCH(1) + EXPBCH(1)*ALOGT - ENEBCH(1)*RTEMP
      ALNKFR    = 0.5*ALNKFR
      ALNKBR    = 0.5*ALNKBR
      AKFB2     = EXP(ALNKFR)
      AKBB2     = EXP(ALNKBR)
      PROD1     = CONCEN(1)*CONCEN(3)
      PROD2     = CONCEN(2)*CONCEN(2)
      OMEGAF    = AKFB2*PROD1*AKFB2
      OMEGAB    = AKBB2*PROD2*AKBB2
      WREACT(1) = OMEGAF - OMEGAB
C
C     FIND NENSPEC FOR THIS REACTION
C
      DENFAC = 0.
      RMIN   = -10.

      IF (WREACT(1) .LT. 0.) THEN
C
C        NENSPEC IS OH
C
         IF (CONCEN(2) .GT. 1.E-6) THEN
            ROM   = 2.*WREACT(1)/CONCEN(2)
            IF (ROM .LT. RMIN) DENFAC = SONDPR/CONCEN(2)*2.*OMEGAB
         ENDIF
C
      ELSE
C
C        NENSPEC IS EITHER H2 OR O2
C
         IF (CONCEN(1) .GT. 1.E-6) THEN
            ROM   = -WREACT(1)/CONCEN(1)
            IF (ROM .LT. RMIN) THEN
                RMIN = RMON
```

```
                DENFAC = SONDPR/CONCEN(1)*OMEGAF
            ENDIF
          ENDIF

            IF (CONCEN(3) .GT. 1.E-6) THEN
               ROM   = -WREACT(1)/CONCEN(3)
               IF (ROM .LT. RMIN) THEN
                   RMIN = RMON
                   DENFAC = SONDPR/CONCEN(3)*OMEGAF
               ENDIF
            ENDIF
C
          ENDIF
C
C         ADJUST THE REACTION CONTRIBUTION FOR NENSPEC
C
          WREACT(1) = WREACT(1)/(1.+DENFAC)
C
C
C         REACTION # 2
C
          ALNKFR    = PREFCH(2) + EXPFCH(2)*ALOGT - ENEFCH(2)*RTEMP
          ALNKBR    = PREBCH(2) + EXPBCH(2)*ALOGT - ENEBCH(2)*RTEMP
          ALNKFR    = 0.5*ALNKFR
          ALNKBR    = 0.5*ALNKBR
          AKFB2     = EXP(ALNKFR)
          AKBB2     = EXP(ALNKBR)
          PROD1     = CONCEN(3)*CONCEN(2)*CONCEN(2)
          PROD2     = CONCEN(4)*CONCEN(4)
          OMEGAF    = AKFB2*PROD1*AKFB2
          OMEGAB    = AKBB2*PROD2*AKBB2
          WREACT(2) = OMEGAF - OMEGAB
C
C         FIND NENSPEC FOR THIS REACTION
C
          RMIN   = -10.
          DENFAC = 0.

          IF (WREACT(2) .LT. 0.) THEN
C
C            NENSPEC IS H2O
C
             IF (CONCEN(4) .GT. 1.E-6) THEN
                 ROM   = 2.*WREACT(2)/CONCEN(4)
                 IF (ROM .LT. RMIN) DENFAC = SONDPR/CONCEN(4)*2.*OMEGAB
             ENDIF
C
          ELSE
C
C            NENSPEC IS EITHER H2 OR OH
C
             IF (CONCEN(2) .GT. 1.E-6) THEN
                 ROM   = -2.*WREACT(2)/CONCEN(2)
                 IF (ROM .LT. RMIN) THEN
                      RMIN = RMON
                      DENFAC = SONDPR/CONCEN(2)*2.*OMEGAF
                 ENDIF
```

```
            ENDIF

            IF (CONCEN(3) .GT. 1.E-6) THEN
                ROM   = -WREACT(2)/CONCEN(3)
                IF (ROM .LT. RMIN) THEN
                    RMIN = RMON
                    DENFAC = SONDPR/CONCEN(3)*OMEGAF
                ENDIF
            ENDIF
C
         ENDIF
C
C        ADJUST THE REACTION CONTRIBUTION FOR NENSPEC
C
         WREACT(2) = WREACT(2)/(1.+DENFAC)
C
C        COMPUTE THE SOURCE TERMS

         BIGWJA(5) =   -AMWTCH(1)*RECWDR* WREACT(1)
         BIGWJA(8) = 2.*AMWTCH(4)*RECWDR* WREACT(2)
         BIGWJA(6) = 2.*AMWTCH(2)*RECWDR*(WREACT(1)-WREACT(2))
         BIGWJA(7) =   -AMWTCH(3)*RECWDR*(WREACT(1)+WREACT(2))

         RETURN
      ENDIF
C
C     COMPUTE THE CONTRIBUTION WREACT TO THE SOURCE TERMS FROM ALL
C     THE REACTIONS

      DO 40 IR = 1, NREACH
         ALNKFR = PREFCH(IR) + EXPFCH(IR)*ALOGT - ENEFCH(IR)*RTEMP
         ALNKBR = PREBCH(IR) + EXPBCH(IR)*ALOGT - ENEBCH(IR)*RTEMP
         ALNKFR = 0.5*ALNKFR
         ALNKBR = 0.5*ALNKBR
         AKFB2  = EXP(ALNKFR)
         AKBB2  = EXP(ALNKBR)
         PROD1  = 1.DO
         PROD2  = 1.DO
         NSRK   = NSRKCH(IR)
         DO 30 IS = 1, NSRK
            ISP   = ITABCH(IS ,IR)
            IP1   = IALOCH(ISP,IR)
            IP2   = IBTOCH(ISP,IR)
            IF (IP1 .NE. 0) PROD1 = PROD1*CONCEN(ISP)**IP1
            IF (IP2 .NE. 0) PROD2 = PROD2*CONCEN(ISP)**IP2
30       CONTINUE
         OMEGAF     = AKFB2*PROD1*AKFB2
         OMEGAB     = AKBB2*PROD2*AKBB2
         WREACT(IR) = OMEGAF - OMEGAB
C
C        FIND NENSPEC FOR THIS REACTION
C
         RMIN   = -10.
         DENFAC = 0.

         DO 35 IS = 1, NSRK
            ISP   = ITABCH(IS ,IR)
```

```
            IF (CONCEN(ISP) .GT. 1.E-6) THEN
               ROM   = BMIACH(IS,IR)*WREACT(IR)/CONCEN(ISP)
               IF (ROM .LT. RMIN) THEN
                  IP1  = IALPCH(ISP,IR)
                  IP2  = IBETCH(ISP,IR)
                  RMIN = RMON
                  DENFAC = SONDPR/CONCEN(ISP)*(IP1*OMEGAF+IP2*OMEGAB)
               ENDIF
            ENDIF
35       CONTINUE
C
C        ADJUST THE REACTION CONTRIBUTION FOR NENSPEC
C
         WREACT(IR) = WREACT(IR)/(1.+DENFAC)
C
40    CONTINUE

C     COMPUTE THE SOURCE TERMS

      DO 60 IS = 1, NEQSCH
         JS     = NEQBAS + IS
         SUMWT  = 0.
         DO 50 IR = 1, NREACH
            SUMWT  = SUMWT + BMIACH(IS,IR)*WREACT(IR)
50       CONTINUE
         BIGWJA(JS) = AMWTCH(IS)*SUMWT
         IF (KROGER .EQ. 2) BIGWJA(JS) = BIGWJA(JS)*RHOD
         BIGWJA(JS) = BIGWJA(JS)*RECWDR
60    CONTINUE

      RETURN
      END
```

G2BPIN

```
      SUBROUTINE G2BPIN (IBNODE,INTERF)

      INCLUDE 'PRECIS.INC '
      INCLUDE 'PARMV2.INC '
      INCLUDE 'G2COMN.INC '
      DIMENSION AGEOMB(4,4), YGEOMB(4), COEFFB(4)
      DIMENSION IBNODE(5)
C
C*********************************************************************
C
C     THIS SUBROUTINE DOES THE INTERPOLATION AT A BOUNDARY NODE FOR A
C     NEWLY DIVIDED CELL.  INTERF INDICATES THE INTERPOLATION FUNCTION
C     TO BE USED FOR THE GEOMETRY (Y-COORDINATE) OF THE NEWLY CREATED
C     NODE ON THE BOUNDARY.  INTERF=1 FOR QUADRATIC, =2 FOR CUBIC AND
C     =3 FOR A CIRCULAR ARC.
C     THE GEOMETRY AT THE FOLLOWING NODES IS KNOWN:
C
C              4       2     0     1       3        IBNODE
```

```
C             ----+--------+-----!-----+-------+-----
C           -     +        +     !     +       +
C            -    +        +     !     +       +
C
C*************************************************************************
C
      X1 = GEOMG2(1,IBNODE(1))
      X2 = GEOMG2(1,IBNODE(2))
      X3 = GEOMG2(1,IBNODE(3))
      X4 = GEOMG2(1,IBNODE(4))
      XO = GEOMG2(1,IBNODE(5))

      Y1 = GEOMG2(2,IBNODE(1))
      Y2 = GEOMG2(2,IBNODE(2))
      Y3 = GEOMG2(2,IBNODE(3))
      Y4 = GEOMG2(2,IBNODE(4))
      YO = GEOMG2(2,IBNODE(5))

      GO TO (100, 100, 300), INTERF
      RETURN
C
C               ------------------
C     SETUP FOR CUBIC INTERPOLATION
C               ------------------
C
100   CONTINUE
      X1    = X1 - XO
      X3    = X3 - XO
      X4    = X4 - XO

      X3    = X3/X1
      X4    = X4/X1
      X3P2  = X3*X3
      X3P3  = X3*X3P2
      X4P2  = X4*X4
      X4P3  = X4*X4P2

      AGEOMB(1,1) = 1.
      AGEOMB(1,2) = 1.
      AGEOMB(1,3) = 1.
      AGEOMB(1,4) = 1.

      AGEOMB(2,1) = 1.
      AGEOMB(2,2) =-1.
      AGEOMB(2,3) = 1.
      AGEOMB(2,4) =-1.

      AGEOMB(3,1) = 1.
      AGEOMB(3,2) = X3
      AGEOMB(3,3) = X3P2
      AGEOMB(3,4) = X3P3

      AGEOMB(4,1) = 1.
      AGEOMB(4,2) = X4
      AGEOMB(4,3) = X4P2
      AGEOMB(4,4) = X4P3
```

```
      YGEOMB(1) = Y1
      YGEOMB(2) = Y2
      YGEOMB(3) = Y3
      YGEOMB(4) = Y4
C
C     QUADRATIC INTERPOLATION WILL BE USED IF THE SECTIONS 4-2 OR
C     1-3 ARE EITHER HORIZONTAL OR VERTICAL LINE SEGMENTS
C
      IQUAD  = 1
      DYTEST = ABS(Y4-Y2)
      IF (DYTEST .LT. 1.E-8) GOTO 200
      DYTEST = ABS(Y3-Y1)
      IF (DYTEST .LT. 1.E-8) THEN
        IQUAD = 2
        GOTO 200
      ENDIF
      DYTEST = ABS(X3-X1)
      IF (DYTEST .LT. 1.E-8) THEN
        IQUAD = 2
        GOTO 200
      ENDIF

      CALL GAUSS2(AGEOMB,YGEOMB,COEFFB,4,4)
C
C     SET THE RESULT FOR CUBIC INTERPOLATION
C
      GEOMG2(2,IBNODE(5)) = COEFFB(1)
      RETURN
C
C               ----------------------
C     SETUP FOR QUADRATIC INTERPOLATION
C               ----------------------
C
200   CONTINUE
C
C     IF (INTERF .NE. 1) RETURN
C
C     SEE IF NODE 4 IS TO BE INSTEAD OF NODE 3
C
      IF (IQUAD .EQ. 2) THEN
        AGEOMB(3,2) = X4
        AGEOMB(3,3) = X4P2
        AGEOMB(3,4) = X4P3
        YGEOMB(3)   = Y4
      ENDIF

      CALL GAUSS2 (AGEOMB,YGEOMB,COEFFB,3,4)
C
C     SET THE RESULT FOR QUADRATIC INTERPOLATION
C
      GEOMG2(2,IBNODE(5)) = COEFFB(1)
      RETURN
C
C               -------------------------
C     SETUP FOR CIRCULAR ARC INTERPOLATION (USE 1,2 AND 4)
C               -------------------------
C
```

```
300     CONTINUE

C       SEE IF POINTS 1, 2, AND 3 WILL BE USED FOR THE ARC

        DYTEST = ABS(Y4-Y2)
        IF (DYTEST .LT. 1.E-8) THEN
          X4 = X3
          Y4 = Y3
        ENDIF

        X12    = X1 - X2
        X42    = X4 - X2
        Y12    = Y1 - Y2
        Y42    = Y4 - Y2
        RR1    = 0.5*(X1*X1 - X2*X2 + Y1*Y1 - Y2*Y2)
        RR2    = 0.5*(X4*X4 - X2*X2 + Y4*Y4 - Y2*Y2)
        YC     = (X42*RR1-X12*RR2)/(X42*Y12-X12*Y42)
        XC     = (RR1-Y12*YC)/X12
        RC2    = (X1-XC)**2 + (Y1-YC)**2
        YRAD   = SQRT ( RC2 - (XO-XC)**2 )
        YPLUS  = YC + YRAD
        YMINUS = YC - YRAD
C
C       SEE WHETHER TO USE POSITIVE OR NEGATIVE SIGN DEPENDING UPON
C       WHICHEVER SOLUTION IS CLOSER TO THE LINEAR ONE
C
        IF ( ABS(YPLUS-YO) .LT. ABS(YMINUS-YO) ) THEN
           JSIGN = 1
           YO    = YPLUS
        ELSE
           JSIGN =-1
           YO    = YMINUS
        ENDIF
C
        X12P2 = X12**2
        Y12P2 = Y12**2
        BIGA  = 1. + Y12P2/X12P2
        BIGB  = 2.*(XC*Y12/X12 - RR1*Y12/X12P2 - YC)
        BIGC  = RR1**2/X12P2 - 2.*RR1*XC/X12 + XC*XC + YC*YC - RC2
        YO    = 0.5*(-BIGB+JSIGN*SQRT(BIGB**2-4.*BIGA*BIGC))/BIGA
        XO    = (RR1-Y12*YO)/X12

        GEOMG2(1,IBNODE(5)) = XO
        GEOMG2(2,IBNODE(5)) = YO

        RETURN
        END
```

G2CLPU

```
        SUBROUTINE  G2CLPO (LSUB1, LSUB2, LSUB3, LSUB4, LCELL, IWARN)
C                   G2CLPU
```

```
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'H2COMN.INC '
      INCLUDE 'HEXCOD.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'M2COMN.INC '

C*********************************************************************

C     THIS SUBROUTINE COLLAPSES THE FOUR SUBCELLS LSUB1, LSUB2, LSUB3,
C     LSUB4 WHICH MAKE UP CELL 'LCELL' AND PERFORMS ALL NECESSARY
C     POINTER SYSTEM REALIGNMENTS

C*********************************************************************

C     FIND THE FOUR CELLS COMPRISING LCELL

      LMSE = 0
      LMNE = 0
      LMSW = MIN (LSUB1, LSUB2, LSUB3, LSUB4)
      LMNW = MAX (LSUB1, LSUB2, LSUB3, LSUB4)

      LDUM = LMSW + 1
      IF (LSUB1 .EQ. LDUM) LMSE = LSUB1
      IF (LSUB2 .EQ. LDUM) LMSE = LSUB2
      IF (LSUB3 .EQ. LDUM) LMSE = LSUB3
      IF (LSUB4 .EQ. LDUM) LMSE = LSUB4

      LDUM = LMNW - 1
      IF (LSUB1 .EQ. LDUM) LMNE = LSUB1
      IF (LSUB2 .EQ. LDUM) LMNE = LSUB2
      IF (LSUB3 .EQ. LDUM) LMNE = LSUB3
      IF (LSUB4 .EQ. LDUM) LMNE = LSUB4
C
C     SEE IF THE GIVEN SUBCELLS LMSW, LMSE, LMNE  & LMNW ARE CONTIGUOUS?
C
      IF (LMSE.NE.(LMSW+1) .OR. LMNE.NE.(LMSW+2)
     1                      .OR. LMNW.NE.(LMSW+3) ) RETURN

C     ------------------------
C     INTERCHNAGE INFORMMATION
C     ------------------------
C     INTERCHANGE (LMSW WITH NLAST1), (LMSE WITH NLAST2)
C                 (LMNE WITH NLAST3), (LMNW WITH NLAST4)

      NLAST4 = NCELG2
      NLAST3 = NLAST4 - 1
      NLAST2 = NLAST3 - 1
      NLAST1 = NLAST2 - 1

C     IF THE CELL TO BE DIVIDED IS ITSELF ONE OF THE LAST CELLS
C     THEN SIMPLY EXIT FOR NOW

      IF (LCELL .GE. NLAST1) RETURN
C
```

```
C       SAVE THE CELL POINTERS
C           -
            -
        KC     = ICELG2(1,LCELL)
        KSW    = ICELG2(2,LCELL)
        KS     = ICELG2(3,LCELL)
        KSE    = ICELG2(4,LCELL)
        KE     = ICELG2(5,LCELL)
        KNE    = ICELG2(6,LCELL)
        KN     = ICELG2(7,LCELL)
        KNW    = ICELG2(8,LCELL)
        KW     = ICELG2(9,LCELL)
        KX     = KAUXG2(LCELL)
        KXSW   = KAUXG2(LMSW)
        KXSE   = KAUXG2(LMSE)
        KXNE   = KAUXG2(LMNE)
        KXNW   = KAUXG2(LMNW)
        K5LMSW = IAND(KXSW,KU000F)
        K5LMSE = IAND(KXSE,KU000F)
        K5LMNE = IAND(KXNE,KU000F)
        K5LMNW = IAND(KXNW,KU000F)
C
C       A CELL WHICH IS PERMANENTLY MARKED FOR THE FUEL INJECTION
C       CAN NOT BE COLLAPSED
C
        IF (IAND(KX,KL2000) .NE. 0) RETURN
C
C       IF THE COMPONENT CELLS ARE BASE CELLS THEN THEY CAN NOT BE
C       COLLAPSED

        IF (K5LMSW .EQ. 0 .OR. K5LMSE .EQ. 0 .OR.
     1      K5LMNE .EQ. 0 .OR. K5LMNW .EQ. 0     ) RETURN

C       FIND THE LEVEL LEVELG OF THE GIVEN CELL LCELL
C       OLD AND NEW LEVELS; LEVELO > 0

        LEVELO = ISHFT(K5LMSW,-16)
        LEVELG = LEVELO - 1
        K5LEVG = IAND (KX,KU000F)

C       IF THE COMPONENT CELLS HAVE DIVIDED NEIGHBOURS THEN
C       THEY CAN NOT BE COLLAPSED (%%%%)

        IF (ICELG2(3,LMSW) .NE. 0) RETURN
        IF (ICELG2(9,LMSW) .NE. 0) RETURN
        IF (ICELG2(3,LMSE) .NE. 0) RETURN
        IF (ICELG2(5,LMSE) .NE. 0) RETURN
        IF (ICELG2(5,LMNE) .NE. 0) RETURN
        IF (ICELG2(7,LMNE) .NE. 0) RETURN
        IF (ICELG2(7,LMNW) .NE. 0) RETURN
        IF (ICELG2(9,LMNW) .NE. 0) RETURN

C
C       FIND CELLS WHICH BOUND DIVIDED CELL
C
C         |------------|--------------|------------|
C         |            |              |            |  K FOR NODE
```

```
C       |      + + + + + + + + + + + + + + + +      |  L FOR CELL
C       |  -   +LCNW | LHNW  + LHNE  |LCNE +      |
C       |  -   +     |       +       |     +      |
C       |------+-----+-------+-------+-----+------|
C       |      +     |KNW   KN    KNE|     +      | M:CENTER (MIDDLE)
C       |      +LVNW | LMNW     LMNE |LVNE +      | C:CORNER (ADJACENT)
C       |      + + + +KW   LCELL   KE+ + + +      | H:HORIZONTAL (ADJAC)
C       |      +LVSW | LMSW     LMSE |LVSE +      | V:VERTICAL (ADJACENT)
C       |      +     |KSW   KS    KSE|     +      |
C       |------+-----+-------+-------+-----+------|
C       |      +     |       +       |     +      |
C       |      +LCSW | LHSW  + LHSE  |LCSE +      |
C       |      + + + + + + + + + + + + + + + +      |
C       |            |               |            |
C       |------------|---------------|------------|
C
        LVSW = NEIBG2(4,KSW)
        LCSW = NEIBG2(1,KSW)
        LHSW = NEIBG2(2,KSW)
        LHSE = NEIBG2(1,KSE)
        LCSE = NEIBG2(2,KSE)
        LVSE = NEIBG2(3,KSE)
        LVNE = NEIBG2(2,KNE)
        LCNE = NEIBG2(3,KNE)
        LHNE = NEIBG2(4,KNE)
        LHNW = NEIBG2(3,KNW)
        LCNW = NEIBG2(4,KNW)
        LVNW = NEIBG2(1,KNW)

C       IF THE COMPONENT CELLS ARE JUST OUTSIDE EMBEDDED REGION THEN
C       THEY CAN NOT BE COLLAPSED; THIS WILL BE SO IF THE LEVELS OF
C       THE NEIGHBOURHOOD CELLS DIFFER BY MORE THAN ONE
C       FIRST DO THE CORNER CELLS

        IF (LCSW .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCSW),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LCSE .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCSE),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LCNE .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCNE),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LCNW .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCNW),KU000F)
```

```
          LEVELC = ISHFT(K6LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF-(IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF
C
C       MARK NODE AT CENTER OF CELL FOR DELETION
C
        DPENG2(1,KC) = -99.
C
C       MARK SOUTHERN NODE FOR DELETION IF NEED BE

        IF (LHSW .EQ. LHSE) THEN
          DPENG2(1,KS) = -99.
          KSS          = KS
          KS           = 0
          LS           = LHSW
        ELSE IF (LHSW .EQ. 0 .OR. LHSE .EQ. 0) THEN
          DPENG2(1,KS) = -99.
          KSS          = KS
          KS           = 0
          LS           = 0
        ELSE
          LS           = ICELG2(10,LHSW)
        ENDIF

C       MARK EASTERN NODE FOR DELETION IF NEED BE

        IF (LVSE .EQ. LVNE) THEN
          DPENG2(1,KE) = -99.
          KEE          = KE
          KE           = 0
          LE           = LVSE
C       ELSE IF (LVNE .EQ. 0 .OR. LVSE .EQ. 0) THEN
C         DPENG2(1,KE) = -99.
C         KEE          = KE
C         KE           = 0
C         LE           = 0
        ELSE
          LE           = ICELG2(10,LVSE)
        ENDIF

C       MARK NORTHERN NODE FOR DELETION IF NEED BE

        IF (LHNE .EQ. LHNW) THEN
          DPENG2(1,KN) = -99.
          KNN          = KN
          KN           = 0
          LN           = LHNE
        ELSE IF (LHNE .EQ. 0 .OR. LHNW .EQ. 0) THEN
          DPENG2(1,KN) = -99.
          KNN          = KN
          KN           = 0
          LN           = 0
        ELSE
          LN           = ICELG2(10,LHNE)
        ENDIF
```

```
C       MARK WESTERN NODE FOR DELETION IF NEED BE

        IF (LVNW .EQ. LVSW) THEN
          DPENG2(1,KW) = -99.
          KWW          = KW
          KW           = 0
          LW           = LVNW
C       ELSE IF (LVNW .EQ. 0 .OR. LVSW .EQ. 0) THEN
C         DPENG2(1,KW) = -99.
C         KWW          = KW
C         KW           = 0
C         LW           = 0
        ELSE
          LW           = ICELG2(10,LVNW)
        ENDIF

C       ----------------------
C       INTERCHNAGE INFORMMATION
C       ----------------------

C       UPDATE NODES (PLUS SUPERCELL) OF THE INTERCHANGED CELLS

        DO 10 IP = 1, 10
            ICELG2(IP,LMSW) = ICELG2(IP,NLAST1)
            ICELG2(IP,LMSE) = ICELG2(IP,NLAST2)
            ICELG2(IP,LMNE) = ICELG2(IP,NLAST3)
            ICELG2(IP,LMNW) = ICELG2(IP,NLAST4)
10      CONTINUE

C       INTERCHANGE THE AUXILLIARY POINTERS

        KAUXG2(LMSW) = KAUXG2(NLAST1)
        KAUXG2(LMSE) = KAUXG2(NLAST2)
        KAUXG2(LMNE) = KAUXG2(NLAST3)
        KAUXG2(LMNW) = KAUXG2(NLAST4)

C       INTERCHANGE THE RECIPROCAL VOLUME POINTERS

        RVOLM2(LMSW) = RVOLM2(NLAST1)
        RVOLM2(LMSE) = RVOLM2(NLAST2)
        RVOLM2(LMNE) = RVOLM2(NLAST3)
        RVOLM2(LMNW) = RVOLM2(NLAST4)

C       INTERCHANGE THE PERIMETER POINTERS

        PERIM2(LMSW) = PERIM2(NLAST1)
        PERIM2(LMSE) = PERIM2(NLAST2)
        PERIM2(LMNE) = PERIM2(NLAST3)
        PERIM2(LMNW) = PERIM2(NLAST4)

C       INTERCHANGE THE METRIC POINTERS

        DXEWM2(LMSW) = DXEWM2(NLAST1)
        DXEWM2(LMSE) = DXEWM2(NLAST2)
        DXEWM2(LMNE) = DXEWM2(NLAST3)
        DXEWM2(LMNW) = DXEWM2(NLAST4)
```

```
      DYEWM2(LMSW) = DYEWM2(NLAST1)
      DYEWM2(LMSE) = DYEWM2(NLAST2)
      DYEWM2(LMNE) = DYEWM2(NLAST3)
      DYEWM2(LMNW) = DYEWM2(NLAST4)

      DXNSM2(LMSW) = DXNSM2(NLAST1)
      DXNSM2(LMSE) = DXNSM2(NLAST2)
      DXNSM2(LMNE) = DXNSM2(NLAST3)
      DXNSM2(LMNW) = DXNSM2(NLAST4)

      DYNSM2(LMSW) = DYNSM2(NLAST1)
      DYNSM2(LMSE) = DYNSM2(NLAST2)
      DYNSM2(LMNE) = DYNSM2(NLAST3)
      DYNSM2(LMNW) = DYNSM2(NLAST4)

C     FIND THE NEIGHBOURS OF THE LAST FOUR CELLS AND
C     CHECK IF CELLS AGREE ON THE NODE ASSIGNMENTS
C
      JC  = ICELG2(6,NLAST1)
      JSW = ICELG2(2,NLAST1)
      JS1 = ICELG2(3,NLAST1)
      JS  = ICELG2(4,NLAST1)
      JS2 = ICELG2(3,NLAST2)
      JSE = ICELG2(4,NLAST2)
      JE1 = ICELG2(5,NLAST2)
      JE  = ICELG2(6,NLAST2)
      JE2 = ICELG2(5,NLAST3)
      JNE = ICELG2(6,NLAST3)
      JN1 = ICELG2(7,NLAST3)
      JN  = ICELG2(8,NLAST3)
      JN2 = ICELG2(7,NLAST4)
      JNW = ICELG2(8,NLAST4)
      JW1 = ICELG2(9,NLAST4)
      JW  = ICELG2(2,NLAST4)
      JW2 = ICELG2(9,NLAST1)
C
C     FIND THE CELLS PERTINENT TO THE ABOVE NODES; SOME OF THESE
C     CELLS MIGHT BE DIVIDED
C
C      JNW    JN2   JN    JN1   JNE       KNW          KN          KNE
C        |-----+-----+-----+-----+         |-----------+-----+-----+
C        |N4NW  N4NE |N3NW  N3NE |         |           |           |
C     JW1+   NLAST4  *   NLAST3  +JE2      |   LMNW    |   LMNE    |
C        |N4SW  N4SE |N3SW  N3SE |         |           |           |
C      JW|-----*----JC-----*-----+JE      KW|----------KC-----------+KN
C        |N1NW  N1NE |N2NW  N2NE |         |           |           |
C     JW2+   NLAST1  *   NLAST2  +JE1      |   LMSW    |   LMSE    |
C        |N1SW  N1SE |N2SW  N2SE |         |           |           |
C        |-----+-----|-----+-----|         |-----------|-----------|
C      JSW    JS1   JS    JS2   JSE       KSW          KS          KSE
C
C
C     INITIALIZE MIDDLE EDGE NODES (INDICATED BY *'S) OF THE LAST
C     FOUR CELLS

      ISTAR1 = 0
      ISTAR2 = 0
```

```
      ISTAR3 = 0
      ISTAR4 = 0
C
      IF (ICELG2(1,NLAST1) .NE. 0) THEN
        ISTAR4 = ICELG2(7,NLAST1)
        ISTAR1 = ICELG2(5,NLAST1)
      ENDIF
C
      IF (ICELG2(1,NLAST2) .NE. 0) THEN
        ISTAR1 = ICELG2(9,NLAST2)
        ISTAR2 = ICELG2(7,NLAST2)
      ENDIF
C
      IF (ICELG2(1,NLAST3) .NE. 0) THEN
        ISTAR2 = ICELG2(3,NLAST3)
        ISTAR3 = ICELG2(9,NLAST3)
      ENDIF
C
      IF (ICELG2(1,NLAST4) .NE. 0) THEN
        ISTAR3 = ICELG2(5,NLAST4)
        ISTAR4 = ICELG2(3,NLAST4)
      ENDIF
C
C     NOW UPDATE THE NEIGHBOURS OF THE INTERCHANGED CELLS
C
      IF (NEIBG2(3,JSW) .EQ. NLAST1) NEIBG2(3,JSW) = LMSW
      IF (NEIBG2(4,JS ) .EQ. NLAST1) NEIBG2(4,JS ) = LMSW
      IF (NEIBG2(1,JC ) .EQ. NLAST1) NEIBG2(1,JC ) = LMSW
      IF (NEIBG2(2,JW ) .EQ. NLAST1) NEIBG2(2,JW ) = LMSW

      IF (NEIBG2(3,JS ) .EQ. NLAST2) NEIBG2(3,JS ) = LMSE
      IF (NEIBG2(4,JSE) .EQ. NLAST2) NEIBG2(4,JSE) = LMSE
      IF (NEIBG2(1,JE ) .EQ. NLAST2) NEIBG2(1,JE ) = LMSE
      IF (NEIBG2(2,JC ) .EQ. NLAST2) NEIBG2(2,JC ) = LMSE

      IF (NEIBG2(3,JC ) .EQ. NLAST3) NEIBG2(3,JC ) = LMNE
      IF (NEIBG2(4,JE ) .EQ. NLAST3) NEIBG2(4,JE ) = LMNE
      IF (NEIBG2(1,JNE) .EQ. NLAST3) NEIBG2(1,JNE) = LMNE
      IF (NEIBG2(2,JN ) .EQ. NLAST3) NEIBG2(2,JN ) = LMNE

      IF (NEIBG2(3,JW ) .EQ. NLAST4) NEIBG2(3,JW ) = LMNW
      IF (NEIBG2(4,JC ) .EQ. NLAST4) NEIBG2(4,JC ) = LMNW
      IF (NEIBG2(1,JN ) .EQ. NLAST4) NEIBG2(1,JN ) = LMNW
      IF (NEIBG2(2,JNW) .EQ. NLAST4) NEIBG2(2,JNW) = LMNW
C
C     UPDATE STAR EDGE POINTS
C
      IF (ISTAR1 .NE. 0) THEN
         IF (NEIBG2(1,ISTAR1) .EQ. NLAST1) NEIBG2(1,ISTAR1) = LMSW
         IF (NEIBG2(4,ISTAR1) .EQ. NLAST1) NEIBG2(4,ISTAR1) = LMSW
         IF (NEIBG2(2,ISTAR1) .EQ. NLAST2) NEIBG2(2,ISTAR1) = LMSE
         IF (NEIBG2(3,ISTAR1) .EQ. NLAST2) NEIBG2(3,ISTAR1) = LMSE
      ENDIF
      IF (ISTAR2 .NE. 0) THEN
         IF (NEIBG2(1,ISTAR2) .EQ. NLAST2) NEIBG2(1,ISTAR2) = LMSE
         IF (NEIBG2(2,ISTAR2) .EQ. NLAST2) NEIBG2(2,ISTAR2) = LMSE
         IF (NEIBG2(3,ISTAR2) .EQ. NLAST3) NEIBG2(3,ISTAR2) = LMNE
```

```
            IF (NEIBG2(4,ISTAR2) .EQ. NLAST3) NEIBG2(4,ISTAR2) = LMNE
         ENDIF
         IF (ISTAR3 .NE. 0) THEN
            IF (NEIBG2(2,ISTAR3) .EQ. NLAST3) NEIBG2(2,ISTAR3) = LMNE
            IF (NEIBG2(3,ISTAR3) .EQ. NLAST3) NEIBG2(3,ISTAR3) = LMNE
            IF (NEIBG2(4,ISTAR3) .EQ. NLAST4) NEIBG2(4,ISTAR3) = LMNW
            IF (NEIBG2(1,ISTAR3) .EQ. NLAST4) NEIBG2(1,ISTAR3) = LMNW
         ENDIF
         IF (ISTAR4 .NE. 0) THEN
            IF (NEIBG2(3,ISTAR4) .EQ. NLAST4) NEIBG2(3,ISTAR4) = LMNW
            IF (NEIBG2(4,ISTAR4) .EQ. NLAST4) NEIBG2(4,ISTAR4) = LMNW
            IF (NEIBG2(1,ISTAR4) .EQ. NLAST1) NEIBG2(1,ISTAR4) = LMSW
            IF (NEIBG2(2,ISTAR4) .EQ. NLAST1) NEIBG2(2,ISTAR4) = LMSW
         ENDIF
C
C        UPDATE THE OTHER NON-ZERO MIDDLE EDGES
C
         IF (JS1 .NE. 0) THEN
            IF (NEIBG2(3,JS1) .EQ. NLAST1) NEIBG2(3,JS1) = LMSW
            IF (NEIBG2(4,JS1) .EQ. NLAST1) NEIBG2(4,JS1) = LMSW
         ENDIF
         IF (JS2 .NE. 0) THEN
            IF (NEIBG2(3,JS2) .EQ. NLAST2) NEIBG2(3,JS2) = LMSE
            IF (NEIBG2(4,JS2) .EQ. NLAST2) NEIBG2(4,JS2) = LMSE
         ENDIF
C
         IF (JE1 .NE. 0) THEN
            IF (NEIBG2(1,JE1) .EQ. NLAST2) NEIBG2(1,JE1) = LMSE
            IF (NEIBG2(4,JE1) .EQ. NLAST2) NEIBG2(4,JE1) = LMSE
         ENDIF
         IF (JE2 .NE. 0) THEN
            IF (NEIBG2(1,JE2) .EQ. NLAST3) NEIBG2(1,JE2) = LMNE
            IF (NEIBG2(4,JE2) .EQ. NLAST3) NEIBG2(4,JE2) = LMNE
         ENDIF
C
         IF (JN1 .NE. 0) THEN
            IF (NEIBG2(1,JN1) .EQ. NLAST3) NEIBG2(1,JN1) = LMNE
            IF (NEIBG2(2,JN1) .EQ. NLAST3) NEIBG2(2,JN1) = LMNE
         ENDIF
         IF (JN2 .NE. 0) THEN
            IF (NEIBG2(1,JN2) .EQ. NLAST4) NEIBG2(1,JN2) = LMNW
            IF (NEIBG2(2,JN2) .EQ. NLAST4) NEIBG2(2,JN2) = LMNW
         ENDIF
C
         IF (JW1 .NE. 0) THEN
            IF (NEIBG2(2,JW1) .EQ. NLAST4) NEIBG2(2,JW1) = LMNW
            IF (NEIBG2(3,JW1) .EQ. NLAST4) NEIBG2(3,JW1) = LMNW
         ENDIF
         IF (JW2 .NE. 0) THEN
            IF (NEIBG2(2,JW2) .EQ. NLAST1) NEIBG2(2,JW2) = LMSW
            IF (NEIBG2(3,JW2) .EQ. NLAST1) NEIBG2(3,JW2) = LMSW
         ENDIF
C
C        IF ANY OF THE LAST FOUR CELLS IS DIVIDED, THEN IT IS THE
C        SUPERCELL OF SOME OTHER CELLS NSONJ AND ITS SUPERCELL
C        WILL HAVE TO BE UPDATED
```

```
      IF (ICELG2(1,NLAST1) .NE. 0 .OR. ICELG2(1,NLAST2) .NE. 0 .OR.
     1    ICELG2(1,NLAST3) .NE. 0 .OR. ICELG2(1,NLAST4) .NE. 0) THEN
          DO 15 NSONJ = ILVLG2(2,0), NCELG2
            ISUP = ICELG2(10,NSONJ)
            IF (ISUP .GE. NLAST1) THEN
              IF (ISUP .EQ. NLAST1) ICELG2(10,NSONJ) = LMSW
              IF (ISUP .EQ. NLAST2) ICELG2(10,NSONJ) = LMSE
              IF (ISUP .EQ. NLAST3) ICELG2(10,NSONJ) = LMNE
              IF (ISUP .EQ. NLAST4) ICELG2(10,NSONJ) = LMNW
            ENDIF
15        CONTINUE
      ENDIF
C
C     ADJUST ANY BOUNDARY CONDITION POINTERS WHICH POINT TO CELLS
C     JUST INTERCHANGED
C
      IF (IAND(KAUXG2(NLAST1),KLOOOF) .NE. 0 .OR.
     1    IAND(KAUXG2(NLAST2),KLOOOF) .NE. 0 .OR.
     2    IAND(KAUXG2(NLAST3),KLOOOF) .NE. 0 .OR.
     3    IAND(KAUXG2(NLAST4),KLOOOF) .NE. 0     ) THEN
          DO 20 IB = 1, NBNDG2
            IF (IBNDG2(2,IB) .GE. NLAST1) THEN
               ND1          = IBNDG2(2,IB) - NLAST1
               IBNDG2(2,IB) = LMSW + ND1
            ENDIF
            IF (IBNDG2(3,IB) .GE. NLAST1) THEN
               ND1          = IBNDG2(3,IB) - NLAST1
               IBNDG2(3,IB) = LMSW + ND1
            ENDIF
20        CONTINUE
      ENDIF
C
C     SET THE NEIGHBOUR-NODE-ARRAY OF THE NEW SUPERCELL
C
      NEIBG2(3,KSW) = LCELL
      NEIBG2(4,KSE) = LCELL
      NEIBG2(1,KNE) = LCELL
      NEIBG2(2,KNW) = LCELL
      NEIBG2(1,KC ) = 0
      NEIBG2(2,KC ) = 0
      NEIBG2(3,KC ) = 0
      NEIBG2(4,KC ) = 0
C
      IF (KS .NE. 0) THEN
        NEIBG2(3,KS)  = LCELL
        NEIBG2(4,KS)  = LCELL
      ELSE
        NEIBG2(1,KSS) = 0
        NEIBG2(2,KSS) = 0
        NEIBG2(3,KSS) = 0
        NEIBG2(4,KSS) = 0
      ENDIF
C
      IF (KE .NE. 0) THEN
        NEIBG2(1,KE)  = LCELL
        NEIBG2(4,KE)  = LCELL
      ELSE
```

```
          NEIBG2(1,KEE) = 0
          NEIBG2(2,KEE) = 0
          NEIBG2(3,KEE) = 0
          NEIBG2(4,KEE) = 0
        ENDIF
C
        IF (KN .NE. 0) THEN
          NEIBG2(1,KN)  = LCELL
          NEIBG2(2,KN)  = LCELL
        ELSE
          NEIBG2(1,KNN) = 0
          NEIBG2(2,KNN) = 0
          NEIBG2(3,KNN) = 0
          NEIBG2(4,KNN) = 0
        ENDIF
C
        IF (KW .NE. 0) THEN
          NEIBG2(2,KW)  = LCELL
          NEIBG2(3,KW)  = LCELL
        ELSE
          NEIBG2(1,KWW) = 0
          NEIBG2(2,KWW) = 0
          NEIBG2(3,KWW) = 0
          NEIBG2(4,KWW) = 0
        ENDIF
C
C       ADJUST TOTAL NUMBER OF CELLS
C
        NCELG2 = NCELG2 - 4
C
C       ADJUST THE MAXIMUM LEVEL IF NEED BE
C
        ILVLG2(3,LEVELO) = ILVLG2(3,LEVELO) - 4
        IF (ILVLG2(3,NLVLG2) .LE. 0) NLVLG2 = NLVLG2 - 1

C       UPDATE THE DIVIDED CELL POINTERS
C
        ICELG2(1,LCELL) = 0
        ICELG2(3,LCELL) = KS
        ICELG2(5,LCELL) = KE
        ICELG2(7,LCELL) = KN
        ICELG2(9,LCELL) = KW

C       RESET EDGE NODE POINTERS OF ALL NEIGHBOURING CELLS

        IF (LS .NE. 0) ICELG2(7,LS) = KS
        IF (LE .NE. 0) ICELG2(9,LE) = KE
        IF (LN .NE. 0) ICELG2(3,LN) = KN
        IF (LW .NE. 0) ICELG2(5,LW) = KW
C
C       SCAN THROUGH ALL BOUNDARY CONDITION POINTERS, LOOKING FOR
C       POINTERS TO THE DIVIDED CELL, SKIP THIS SECTION IF LCELL IS
C       NOT A BOUNDARY CELL

        ITWO = 0
        IMD2 = 0
        ICEN = 0
```

```
      IMD1 = 0
      IONE = 0
C
C
C               320        300        280
C                   +-------+-------+
C                   |13D   12C   14E|
C               340 +9   KAUXG2    6+ 260
C                   |11B    3      7|        360 : ERROR
C                   +-------+-------+
C               200        220        240    GO TO STATEMENTS
C
C     BRANCH OUT DEPENDING ON BOUNDARY TYPE
C     (3,S),(6,E),(7,SE),(9,W),(11,SW),(12,N),(13,NW),(14,NE)
C
      IGOTO = IAND (KX,KLOOOF) + 1
      GOTO (370, 360, 360, 220, 360, 360, 260, 240,
     1       360, 340, 360, 200, 300, 320, 280, 360), IGOTO

C
C     SOUTHWESTERN CORNER
C
200   DO 210 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KNW) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KWW) IMD1 = IB
        IF (IBNDG2(1,IB) .EQ. KSW) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KSS) IMD2 = IB
        IF (IBNDG2(1,IB) .EQ. KSE) ITWO = IB
210   CONTINUE
C
C     CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
C
      NBCPG2(1,1) = IONE
      NBCPG2(1,2) = ITWO
      GO TO 368
C
C     SOUTHERN SIDE
C
220   DO 230 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSW) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KSS) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KSE) ITWO = IB
230   CONTINUE
      GOTO 369
C
C     SOUTHEASTERN CORNER
C
240   DO 250 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSW) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KSS) IMD1 = IB
        IF (IBNDG2(1,IB) .EQ. KSE) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KEE) IMD2 = IB
        IF (IBNDG2(1,IB) .EQ. KNE) ITWO = IB
250   CONTINUE
C
C     CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
C
```

```
      NBCPG2(2,1) = IONE
      NBCPG2(2,2) = ITWO
      GO TO 368
C
C     EASTERN SIDE
C
260   DO 270 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSE) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KEE) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KNE) ITWO = IB
270   CONTINUE
      GOTO 369
C
C     NORTHEASTERN CORNER
C
280   DO 290 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSE) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KEE) IMD1 = IB
        IF (IBNDG2(1,IB) .EQ. KNE) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KNN) IMD2 = IB
        IF (IBNDG2(1,IB) .EQ. KNW) ITWO = IB
290   CONTINUE
C
C     CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
C
      NBCPG2(3,1) = IONE
      NBCPG2(3,2) = ITWO
      GO TO 368
C
C     NORTHERN SIDE
C
300   DO 310 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KNE) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KNN) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KNW) ITWO = IB
310   CONTINUE
      GOTO 369
C
C     NORTHWESTERN CORNER
C
320   DO 330 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KNE) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KNN) IMD1 = IB
        IF (IBNDG2(1,IB) .EQ. KNW) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KWW) IMD2 = IB
        IF (IBNDG2(1,IB) .EQ. KSW) ITWO = IB
330   CONTINUE
C
C     CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
C
      NBCPG2(4,1) = IONE
      NBCPG2(4,2) = ITWO
      GO TO 368
C
C     WESTERN SIDE
C
340   DO 350 IB = 1, NBNDG2
```

```
          IF (IBNDG2(1,IB) .EQ. KNW) IONE = IB
          IF (IBNDG2(1,IB) .EQ. KWW) ICEN = IB
          IF (IBNDG2(1,IB) .EQ. KSW) ITWO = IB
350       CONTINUE
          GOTO 369
C
C         CHECK THE EDGE CELLS

368       IF (IONE .EQ. 0 .OR. IMD1 .EQ. 0 .OR. ICEN .EQ. 0
     1                    .OR. IMD2 .EQ. 0 .OR. ITWO .EQ. 0) GOTO 360
C
C         MARK FOR DELETE

          IBNDG2(1,IMD1) = -9
          IBNDG2(1,IMD2) = -9

C         REASSIGN POINTERS

          IBNDG2(3,IONE) = LCELL
          IBNDG2(2,ICEN) = LCELL
          IBNDG2(2,ITWO) = LCELL

          GOTO 370
C
C         CHECK THE EDGE NODES

C369      IF (IONE .EQ. 0 .OR. ICEN .EQ. 0 .OR. ITWO .EQ. 0) GOTO 360
369       IF (ICEN .EQ. 0) GOTO 360
C
C         MARK FOR DELETE

          IBNDG2(1,ICEN) = -9

C         REASSIGN POINTERS

c         IBNDG2(3,IONE) = LCELL
          if (ibndg2(3,ione) .ne. 0) then
             IBNDG2(3,IONE) = LCELL
          else
             IBNDG2(2,IONE) = LCELL
          endif
          IBNDG2(2,ITWO) = LCELL

          GOTO 370
C
C         ERROR IN BOUNDARY CELL POINTERS

360       ZER1 = LCELL
          ZER2 = IGOTO
          CALL ERRORM (17,'G2CLPO','LCELL ',ZER1,'IGOTO ',ZER2,JPRINT,
     1         'ERROR IN BOUNDARY NODE CALCULATION')

370       CONTINUE
C
C         CHECK IF THE CELL HAS FUEL INJECTED TO IT
C
          IF (IAND(KX,KL1000) .EQ. 0) RETURN
```

```
      KUMDH2 = 0

      DO 380 IB = 1, NUMDH2
        IF (NODEH2(IB) .EQ. Kee) THEN
           IBHERE = IB
           GOTO 390
        ENDIF
380   CONTINUE

390   DO 400 IB = IBHERE, NUMDH2-1
        NODEH2(IB) = NODEH2(IB+1)
400   CONTINUE

      NUMDH2 = NUMDH2 - 1

      RETURN
      END
```

G2CLP0

```
      SUBROUTINE  G2CLP0 (LSUB1, LSUB2, LSUB3, LSUB4, LCELL, IWARN)

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.INC     '
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      LOGICAL  IWRITE

C*********************************************************************

C     THIS SUBROUTINE COLLAPSES THE FOUR SUBCELLS LSUB1, LSUB2, LSUB3,
C     LSUB4 WHICH MAKE UP CELL 'LCELL' AND PERFORMS ALL NECESSARY
C     POINTER SYSTEM REALIGNMENTS

C*********************************************************************

      MPOINT = 10
      NADCEL = 4

C     ---------------
C     ERROR CONDITIONS
C     ---------------

C     FIND THE FOUR CELLS COMPRISING LCELL

      LMSE = 0
      LMNE = 0
      LMSW = MIN (LSUB1, LSUB2, LSUB3, LSUB4)
      LMNW = MAX (LSUB1, LSUB2, LSUB3, LSUB4)

      LDUM = LMSW + 1
```

```
      IF (LSUB1 .EQ. LDUM) LMSE = LSUB1
      IF (LSUB2 .EQ. LDUM) LMSE = LSUB2
      IF (LSUB3 .EQ. LDUM) LMSE = LSUB3
      IF (LSUB4 .EQ. LDUM) LMSE = LSUB4

      LDUM = LMNW - 1
      IF (LSUB1 .EQ. LDUM) LMNE = LSUB1
      IF (LSUB2 .EQ. LDUM) LMNE = LSUB2
      IF (LSUB3 .EQ. LDUM) LMNE = LSUB3
      IF (LSUB4 .EQ. LDUM) LMNE = LSUB4
C
C     SEE IF THE GIVEN SUBCELLS LMSW, LMSE, LMNE  & LMNW ARE CONTIGUOUS?
C
      IF (LMSE.NE.(LMSW+1) .OR. LMNE.NE.(LMSW+2)
     1                      .OR. LMNW.NE.(LMSW+3) ) THEN
        ZER1 = LMSW
        ZER2 = LMNW
        CALL WARNIN (13,'G2CLPO','LMSW  ',ZER1,'LMNW  ',ZER2,JPRINT,
     1     'THE GIVEN FINE CELLS ARE NOT CONTIGUOUS')
        IWARN = 13
      ENDIF
C
C     CHECK IF THE FOUR BASE CELLS HAVE THE SAME SUPERCELL LCELL
C
      IF (LCELL.NE.ICELG2(10,LMSW) .OR. LCELL.NE.ICELG2(10,LMSE) .OR.
     1    LCELL.NE.ICELG2(10,LMNE) .OR. LCELL.NE.ICELG2(10,LMNW)) THEN
        ZER1 = LCELL
        ZER2 = LMSW
        CALL ERRORM (14,'G2CLPO','LCELL ',ZER1,'LMSW  ',ZER2,JPRINT,
     1     'THE SUBCELLS DO NOT HAVE SUPERCELL LCELL')
      ENDIF
C
C     ------------------------
C     INTERCHNAGE INFORMMATION
C     ------------------------
C     INTERCHANGE (LMSW WITH NLAST1), (LMSE WITH NLAST2)
C                 (LMNE WITH NLAST3), (LMNW WITH NLAST4)

      NLAST4 = NCELG2
      NLAST3 = NLAST4 - 1
      NLAST2 = NLAST3 - 1
      NLAST1 = NLAST2 - 1

C     IF THE CELL TO BE DIVIDED IS ITSELF ONE OF THE LAST CELLS
C     THEN SIMPLY EXIT

      IF (LCELL .GE. NLAST1) RETURN
C
C     SAVE THE CELL POINTERS
C
      KC    = ICELG2(1,LCELL)
      KSW   = ICELG2(2,LCELL)
      KS    = ICELG2(3,LCELL)
      KSE   = ICELG2(4,LCELL)
      KE    = ICELG2(5,LCELL)
      KNE   = ICELG2(6,LCELL)
      KN    = ICELG2(7,LCELL)
```

```
      KNW    = ICELG2(8,LCELL)
      KW  -  = ICELG2(9,LCELL)
      KX  -  = KAUXG2(LCELL)
      KXSW   = KAUXG2(LMSW)
      KXSE   = KAUXG2(LMSE)
      KXNE   = KAUXG2(LMNE)
      KXNW   = KAUXG2(LMNW)
      K5LMSW = IAND(KXSW,KU000F)
      K5LMSE = IAND(KXSE,KU000F)
      K5LMNE = IAND(KXNE,KU000F)
      K5LMNW = IAND(KXNW,KU000F)
C
C     A CELL WHICH IS PERMANENTLY MARKED FOR THE FUEL INJECTION
C     CAN NOT BE COLLAPSED
C
      IF (IAND(KX,KL2000) .NE. 0) RETURN

C     IF THE COMPONENT CELLS ARE BASE CELLS THEN THEY CAN NOT BE
C     COLLAPSED

      IF (K5LMSW .EQ. 0 .OR. K5LMSE .EQ. 0 .OR.
     1     K5LMNE .EQ. 0 .OR. K5LMNW .EQ. 0     ) RETURN
C
C     CHECK SOME OF THE NODE ASSIGNMENTS
C
      IF (KSW .NE. ICELG2(2,LMSW)) THEN
        ZER1 = KSW
        ZER2 = LMSW
        CALL ERRORM (12,'G2CLPO','KSW   ',ZER1,'LMSW  ',ZER2,JPRINT,
     1       'ERROR IN NODE ASSIGNMENT')
      ENDIF
C
      IF (KSE .NE. ICELG2(4,LMSE)) THEN
        ZER1 = KSE
        ZER2 = LMSE
        CALL ERRORM (12,'G2CLPO','KSE   ',ZER1,'LMSE  ',ZER2,JPRINT,
     1       'ERROR IN NODE ASSIGNMENT')
      ENDIF
C
      IF (KNE .NE. ICELG2(6,LMNE)) THEN
        ZER1 = KNE
        ZER2 = LMNE
        CALL ERRORM (12,'G2CLPO','KNE   ',ZER1,'LMNE  ',ZER2,JPRINT,
     1       'ERROR IN NODE ASSIGNMENT')
      ENDIF
C
      IF (KNW .NE. ICELG2(8,LMNW)) THEN
        ZER1 = KNW
        ZER2 = LMNW
        CALL ERRORM (12,'G2CLPO','KNW   ',ZER1,'LMNW  ',ZER2,JPRINT,
     1       'ERROR IN NODE ASSIGNMENT')
      ENDIF

C     FIND THE LEVEL LEVELG OF THE GIVEN CELL LCELL
C     OLD AND NEW LEVELS; LEVELO > 0

      LEVELO = ISHFT(K5LMSW,-16)
```

```
      LEVELG = LEVELO - 1
      K5LEVG = IAND (KX,KUOOOF)
          -
C     CHECK IF LEVELG=ISHFT(K5LEVG,-16)

      IF (LEVELG .NE. ISHFT(K5LEVG,-16)) THEN
        ZER1 = LEVELG
        ZER2 = ISHFT(K5LEVG,-16)
        CALL ERRORM (15,'G2CLPO','LEVELG',ZER1,'LEVELC',ZER2,
     1      'ERROR IN LEVEL CALCULATION')
      ENDIF

C     IF THE COMPONENT CELLS HAVE DIVIDED NEIGHBOURS THEN
C     THEY CAN NOT BE COLLAPSED (%%%%)

      IF (ICELG2(3,LMSW) .NE. 0) RETURN
      IF (ICELG2(9,LMSW) .NE. 0) RETURN
      IF (ICELG2(3,LMSE) .NE. 0) RETURN
      IF (ICELG2(5,LMSE) .NE. 0) RETURN
      IF (ICELG2(5,LMNE) .NE. 0) RETURN
      IF (ICELG2(7,LMNE) .NE. 0) RETURN
      IF (ICELG2(7,LMNW) .NE. 0) RETURN
      IF (ICELG2(9,LMNW) .NE. 0) RETURN

C
C     FIND CELLS WHICH BOUND DIVIDED CELL
C
C       |------------|--------------|------------|
C       |            |              |            |  K FOR NODE
C       |      + + + + + + + + + + + + + +      |  L FOR CELL
C       |      +LCNW | LHNW  + LHNE  |LCNE +      |
C       |      +     |       +       |     +      |
C       |------+-----+-------+-------+-----+------|
C       |      +     |KNW   KN    KNE|     +      | M:CENTER (MIDDLE)
C       |      +LVNW | LMNW     LMNE |LVNE +      | C:CORNER (ADJACENT)
C       |      + + + +KW   LCELL   KE+ + + +      | H:HORIZONTAL (ADJAC)
C       |      +LVSW | LMSW     LMSE |LVSE +      | V:VERTICAL (ADJACENT)
C       |      +     |KSW   KS    KSE|     +      |
C       |------+-----+-------+-------+-----+------|
C       |      +     |       +       |     +      |
C       |      +LCSW | LHSW  + LHSE  |LCSE +      |
C       |      + + + + + + + + + + + + + +      |
C       |            |              |            |
C       |------------|--------------|------------|
C

      LVSW = NEIBG2(4,KSW)
      LCSW = NEIBG2(1,KSW)
      LHSW = NEIBG2(2,KSW)
      LHSE = NEIBG2(1,KSE)
      LCSE = NEIBG2(2,KSE)
      LVSE = NEIBG2(3,KSE)
      LVNE = NEIBG2(2,KNE)
      LCNE = NEIBG2(3,KNE)
      LHNE = NEIBG2(4,KNE)
      LHNW = NEIBG2(3,KNW)
      LCNW = NEIBG2(4,KNW)
      LVNW = NEIBG2(1,KNW)
```

```
C       IF THE COMPONENT CELLS ARE JUST OUTSIDE EMBEDDED REGION THEN
C       THEY CAN NOT BE COLLAPSED; THIS WILL BE SO IF THE LEVELS OF
C       THE NEIGHBOURHOOD CELLS DIFFER BY MORE THAN ONE
C       FIRST DO THE CORNER CELLS

        IF (LCSW .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCSW),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LCSE .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCSE),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LCNE .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCNE),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LCNW .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCNW),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF
C
C       NOW DO EDGE CELLS
C       ******  THIS IS PROBABLY NOT NEEDED DUE TO (%%%%)  ******
C
        IF (LHSW .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LHSW),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LVSE .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LVSE),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF

        IF (LHNE .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LHNE),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDLC   = LEVELC-LEVELG
          IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
        ENDIF
```

```
      IF (LVNW .NE. 0) THEN
        K5LEDG = IAND(KAUXG2(LVNW),KU000F)
        LEVELC = ISHFT(K5LEDG,-16)
        IDLC   = LEVELC-LEVELG
        IF (IDLC .LT. 0 .OR. IDLC .GT. 1) RETURN
      ENDIF
C
C     -----------
C     DEBUG PRINT
C     -----------
C
C     PRINT OUT PARAMETERS BEFORE CELL MERGER
C
      IWRITE  = IDBGG2 .EQ. 3 .OR. IDBGG2 .GT. 1000
C
      IF (IWRITE)  THEN
         WRITE(JDEBUG,1000)
         WRITE(JDEBUG,1100)
         WRITE(JDEBUG,1200)
         WRITE(JDEBUG,1300)
C
C        GENERAL INFORMMATION
C
         WRITE(JDEBUG,1400) NNODG2, NCELG2, NBNDG2, LEVELG
C
C        POINTERS OF MAIN CELL LCELL
C
         WRITE(JDEBUG,1500) LCELL, KC , KSW, KS , KSE, KE,
     1                              KNE, KN , KNW, KW , KX
         WRITE(JDEBUG,1600) (ICELG2(I,LCELL),I=1,10),KAUXG2(LCELL)
C
C        CELLS TO BE DESTROYED (REASSIGNED)
C
         WRITE(JDEBUG,1700) LMSW, LMSE, LMNE, LMNW
         WRITE(JDEBUG,1800) (ICELG2(I,LMSW),I=1,10),KAUXG2(LMSW)
         WRITE(JDEBUG,1900) (ICELG2(I,LMSE),I=1,10),KAUXG2(LMSE)
         WRITE(JDEBUG,2000) (ICELG2(I,LMNE),I=1,10),KAUXG2(LMNE)
         WRITE(JDEBUG,2100) (ICELG2(I,LMNW),I=1,10),KAUXG2(LMNW)
C
C        NEIGHBOUR CELLS AND THEIR POINTERS
C
         WRITE(JDEBUG,2200) LVSW, LCSW, LHSW, LHSE, LCSE, LVSE,
     1                      LVNE, LCNE, LHNE, LHNW, LCNW, LVNW
C
         IF (LVSW .NE. 0) THEN
            WRITE(JDEBUG,2300) (ICELG2(I,LVSW),I=1,10),KAUXG2(LVSW)
         ENDIF
         IF (LCSW .NE. 0) THEN
            WRITE(JDEBUG,2400) (ICELG2(I,LCSW),I=1,10),KAUXG2(LCSW)
         ENDIF
         IF (LHSW .NE. 0) THEN
            WRITE(JDEBUG,2500) (ICELG2(I,LHSW),I=1,10),KAUXG2(LHSW)
         ENDIF
C
         IF (LHSE .NE. 0) THEN
            WRITE(JDEBUG,2600) (ICELG2(I,LHSE),I=1,10),KAUXG2(LHSE)
```

```
            ENDIF
            IF (LCSE .NE. 0) THEN
             - WRITE(JDEBUG,2700) (ICELG2(I,LCSE),I=1,10),KAUXG2(LCSE)
            ENDIF
            IF (LVSE .NE. 0) THEN
               WRITE(JDEBUG,2800) (ICELG2(I,LVSE),I=1,10),KAUXG2(LVSE)
            ENDIF
C
            IF (LVNE .NE. 0) THEN
               WRITE(JDEBUG,2900) (ICELG2(I,LVNE),I=1,10),KAUXG2(LVNE)
            ENDIF
            IF (LCNE .NE. 0) THEN
               WRITE(JDEBUG,3000) (ICELG2(I,LCNE),I=1,10),KAUXG2(LCNE)
            ENDIF
            IF (LHNE .NE. 0) THEN
               WRITE(JDEBUG,3100) (ICELG2(I,LHNE),I=1,10),KAUXG2(LHNE)
            ENDIF
C
            IF (LHNW .NE. 0) THEN
               WRITE(JDEBUG,3200) (ICELG2(I,LHNW),I=1,10),KAUXG2(LHNW)
            ENDIF
            IF (LCNW .NE. 0) THEN
               WRITE(JDEBUG,3300) (ICELG2(I,LCNW),I=1,10),KAUXG2(LCNW)
            ENDIF
            IF (LVNW .NE. 0) THEN
               WRITE(JDEBUG,3400) (ICELG2(I,LVNW),I=1,10),KAUXG2(LVNW)
            ENDIF
C
C           NEIGHBOURING CELLS OF ALL NODES OF LCELL
C
            WRITE(JDEBUG,3500) (NEIBG2(I,KC ),I=1,4)
            WRITE(JDEBUG,3600) (NEIBG2(I,KSW),I=1,4)
            WRITE(JDEBUG,3700) (NEIBG2(I,KS ),I=1,4)
            WRITE(JDEBUG,3800) (NEIBG2(I,KSE),I=1,4)
            WRITE(JDEBUG,3900) (NEIBG2(I,KE ),I=1,4)
            WRITE(JDEBUG,4000) (NEIBG2(I,KNE),I=1,4)
            WRITE(JDEBUG,4100) (NEIBG2(I,KN ),I=1,4)
            WRITE(JDEBUG,4200) (NEIBG2(I,KNW),I=1,4)
            WRITE(JDEBUG,4300) (NEIBG2(I,KW ),I=1,4)
C
         ENDIF       ! IWRITE
C
C        MARK NODE AT CENTER OF CELL FOR DELETION
C
         DPENG2(1,KC) = -99.
C
C        MARK SOUTHERN NODE FOR DELETION IF NEED BE

         IF (LHSW .EQ. LHSE) THEN
           DPENG2(1,KS) = -99.
           KSS          = KS
           KS           = 0
           LS           = LHSW
         ELSE IF (LHSW .EQ. 0 .OR. LHSE .EQ. 0) THEN
           DPENG2(1,KS) = -99.
           KSS          = KS
           KS           = 0
```

```
          LS            = 0
        ELSE
          LS-           = ICELG2(10,LHSW)
        ENDIF

C       MARK EASTERN NODE FOR DELETION IF NEED BE

        IF (LVSE .EQ. LVNE) THEN
          DPENG2(1,KE) = -99.
          KEE           = KE
          KE            = 0
          LE            = LVSE
C       ELSE IF (LVNE .EQ. 0 .OR. LVSE .EQ. 0) THEN
C         DPENG2(1,KE) = -99.
C         KEE           = KE
C         KE            = 0
C         LE            = 0
        ELSE
          LE            = ICELG2(10,LVSE)
        ENDIF

C       MARK NORTHERN NODE FOR DELETION IF NEED BE

        IF (LHNE .EQ. LHNW) THEN
          DPENG2(1,KN) = -99.
          KNN           = KN
          KN            = 0
          LN            = LHNE
        ELSE IF (LHNE .EQ. 0 .OR. LHNW .EQ. 0) THEN
          DPENG2(1,KN) = -99.
          KNN           = KN
          KN            = 0
          LN            = 0
        ELSE
          LN            = ICELG2(10,LHNE)
        ENDIF

C       MARK WESTERN NODE FOR DELETION IF NEED BE

        IF (LVNW .EQ. LVSW) THEN
          DPENG2(1,KW) = -99.
          KWW           = KW
          KW            = 0
          LW            = LVNW
C       ELSE IF (LVNW .EQ. 0 .OR. LVSW .EQ. 0) THEN
C         DPENG2(1,KW) = -99.
C         KWW           = KW
C         KW            = 0
C         LW            = 0
        ELSE
          LW            = ICELG2(10,LVNW)
        ENDIF

C       ------------------------
C       INTERCHNAGE INFORMMATION
C       ------------------------
```

```
C       UPDATE NODES (PLUS SUPERCELL) OF THE INTERCHANGED CELLS

        DO 10 IP = 1, MPOINT
            ICELG2(IP,LMSW) = ICELG2(IP,NLAST1)
            ICELG2(IP,LMSE) = ICELG2(IP,NLAST2)
            ICELG2(IP,LMNE) = ICELG2(IP,NLAST3)
            ICELG2(IP,LMNW) = ICELG2(IP,NLAST4)
10      CONTINUE

        KAUXG2(LMSW) = KAUXG2(NLAST1)
        KAUXG2(LMSE) = KAUXG2(NLAST2)
        KAUXG2(LMNE) = KAUXG2(NLAST3)
        KAUXG2(LMNW) = KAUXG2(NLAST4)

C       INTERCHANGE THE RECIPROCAL VOLUME POINTERS

        RVOLM2(LMSW) = RVOLM2(NLAST1)
        RVOLM2(LMSE) = RVOLM2(NLAST2)
        RVOLM2(LMNE) = RVOLM2(NLAST3)
        RVOLM2(LMNW) = RVOLM2(NLAST4)

C       INTERCHANGE THE PERIMETER POINTERS

        PERIM2(LMSW) = PERIM2(NLAST1)
        PERIM2(LMSE) = PERIM2(NLAST2)
        PERIM2(LMNE) = PERIM2(NLAST3)
        PERIM2(LMNW) = PERIM2(NLAST4)

C       INTERCHANGE THE METRIC POINTERS

        DXEWM2(LMSW) = DXEWM2(NLAST1)
        DXEWM2(LMSE) = DXEWM2(NLAST2)
        DXEWM2(LMNE) = DXEWM2(NLAST3)
        DXEWM2(LMNW) = DXEWM2(NLAST4)

        DYEWM2(LMSW) = DYEWM2(NLAST1)
        DYEWM2(LMSE) = DYEWM2(NLAST2)
        DYEWM2(LMNE) = DYEWM2(NLAST3)
        DYEWM2(LMNW) = DYEWM2(NLAST4)

        DXNSM2(LMSW) = DXNSM2(NLAST1)
        DXNSM2(LMSE) = DXNSM2(NLAST2)
        DXNSM2(LMNE) = DXNSM2(NLAST3)
        DXNSM2(LMNW) = DXNSM2(NLAST4)

        DYNSM2(LMSW) = DYNSM2(NLAST1)
        DYNSM2(LMSE) = DYNSM2(NLAST2)
        DYNSM2(LMNE) = DYNSM2(NLAST3)
        DYNSM2(LMNW) = DYNSM2(NLAST4)
C
C       FIND THE NEIGHBOURS OF THE LAST FOUR CELLS AND
C       CHECK IF CELLS AGREE ON THE NODE ASSIGNMENTS
C
        JC  = ICELG2(6,NLAST1)
        JD1 = ICELG2(8,NLAST2)
        JD2 = ICELG2(2,NLAST3)
        JD3 = ICELG2(4,NLAST4)
```

```
C
      IF (JC .NE. JD1 .OR. JC .NE. JD2 .OR. JC .NE. JD3) THEN
        ZER1 = NLAST1
        ZER2 = NLAST2
        CALL ERRORM (16,'G2CLPO','NLAST1',ZER1,'NLAST2',ZER2,JPRINT,
     1       'ERROR IN CENTER NODE ASSIGNMENT OF THE LAST FOUR CELLS')
      ENDIF

      JSW = ICELG2(2,NLAST1)
      JS1 = ICELG2(3,NLAST1)
      JS  = ICELG2(4,NLAST1)
      JS2 = ICELG2(3,NLAST2)
      JD1 = ICELG2(2,NLAST2)
      JSE = ICELG2(4,NLAST2)

      IF (JS .NE. JD1) THEN
        ZER1 = NLAST1
        ZER2 = NLAST2
        CALL ERRORM (16,'G2CLPO','NLAST1',ZER1,'NLAST2',ZER2,JPRINT,
     1       'ERROR IN SOUTH NODE ASSIGNMENT OF THE LAST FOUR CELLS')
      ENDIF

      JE1 = ICELG2(5,NLAST2)
      JE  = ICELG2(6,NLAST2)
      JE2 = ICELG2(5,NLAST3)
      JD1 = ICELG2(4,NLAST3)

      IF (JE .NE. JD1) THEN
        ZER1 = NLAST2
        ZER2 = NLAST3
        CALL ERRORM (16,'G2CLPO','NLAST2',ZER1,'NLAST3',ZER2,JPRINT,
     1       'ERROR IN EAST NODE ASSIGNMENT OF THE LAST FOUR CELLS')
      ENDIF

      JNE = ICELG2(6,NLAST3)
      JN1 = ICELG2(7,NLAST3)
      JN  = ICELG2(8,NLAST3)
      JN2 = ICELG2(7,NLAST4)
      JD1 = ICELG2(6,NLAST4)
      JNW = ICELG2(8,NLAST4)

      IF (JN .NE. JD1) THEN
        ZER1 = NLAST3
        ZER2 = NLAST4
        CALL ERRORM (16,'G2CLPO','NLAST3',ZER1,'NLAST4',ZER2,JPRINT,
     1       'ERROR IN NORTH NODE ASSIGNMENT OF THE LAST FOUR CELLS')
      ENDIF

      JW1 = ICELG2(9,NLAST4)
      JW  = ICELG2(2,NLAST4)
      JW2 = ICELG2(9,NLAST1)
      JD1 = ICELG2(8,NLAST1)

      IF (JW .NE. JD1) THEN
        ZER1 = NLAST4
        ZER2 = NLAST1
        CALL ERRORM (16,'G2CLPO','NLAST4',ZER1,'NLAST1',ZER2,JPRINT,
```

```
     1       'ERROR IN WEST NODE ASSIGNMENT OF THE LAST FOUR CELLS')
           ENDIF
C          -
C
C
C          FIND THE CELLS PERTINENT TO THE ABOVE NODES; SOME OF THESE
C          CELLS MIGHT BE DIVIDED
C
C         JNW    JN2   JN    JN1   JNE       KNW          KN          KNE
C           |-----+-----+-----+-----+          |-----------+-----+-----+
C           |N4NW  N4NE |N3NW  N3NE |          |           |           |
C         JW1+   NLAST4  *   NLAST3  +JE2      |   LMNW    |   LMNE    |
C           |N4SW  N4SE |N3SW  N3SE |          |           |           |
C          JW|-----*----JC-----*-----+JE      KW|----------KC----------+KN
C           |N1NW  N1NE |N2NW  N2NE |          |           |           |
C         JW2+   NLAST1  *   NLAST2  +JE1      |   LMSW    |   LMSE    |
C           |N1SW  N1SE |N2SW  N2SE |          |           |           |
C           |-----+-----|-----+-----|          |-----------|-----------|
C          JSW    JS1   JS    JS2   JSE       KSW          KS          KSE
C
C
C          INITIALIZE MIDDLE EDGE NODES (INDICATED BY *'S) OF THE LAST
C          FOUR CELLS

           ISTAR1 = 0
           ISTAR2 = 0
           ISTAR3 = 0
           ISTAR4 = 0
C
C          IF THE LAST FOUR CELLS ARE DIVIDED, THEN ABOVE CELLS ARE

           IF (ICELG2(1,NLAST1) .NE. 0) THEN
             ISTAR4 = ICELG2(7,NLAST1)
             ISTAR1 = ICELG2(5,NLAST1)
           ENDIF
C
           IF (ICELG2(1,NLAST2) .NE. 0) THEN
             ISTAR1 = ICELG2(9,NLAST2)
             ISTAR2 = ICELG2(7,NLAST2)
           ENDIF
C
           IF (ICELG2(1,NLAST3) .NE. 0) THEN
             ISTAR2 = ICELG2(3,NLAST3)
             ISTAR3 = ICELG2(9,NLAST3)
           ENDIF
C
           IF (ICELG2(1,NLAST4) .NE. 0) THEN
             ISTAR3 = ICELG2(5,NLAST4)
             ISTAR4 = ICELG2(3,NLAST4)
           ENDIF
C
C          -----------
C          DEBUG PRINT
C          -----------
C
C          PRINT OUT PARAMETERS BEFORE CELL MERGER OF LAST FOUR CELLS
C
           IF (IWRITE)  THEN
```

```
C
C          POINTERS OF MAIN LAST FOUR CELLS
C           -
           WRITE(JDEBUG,4400) NLAST1, NLAST2, NLAST3, NLAST4
           WRITE(JDEBUG,4500) JC , JSW, JS , JSE, JE, JNE, JN, JNW, JW,
     1                        ISTAR1, ISTAR2, ISTAR3, ISTAR4
           WRITE(JDEBUG,4600)1,(ICELG2(I,NLAST1),I=1,10),KAUXG2(NLAST1)
           WRITE(JDEBUG,4600)2,(ICELG2(I,NLAST2),I=1,10),KAUXG2(NLAST2)
           WRITE(JDEBUG,4600)3,(ICELG2(I,NLAST3),I=1,10),KAUXG2(NLAST3)
           WRITE(JDEBUG,4600)4,(ICELG2(I,NLAST4),I=1,10),KAUXG2(NLAST4)
C
C          NEIGHBOURING CELLS OF ALL NODES OF LAST FOUR CELLS
C
           WRITE(JDEBUG,4700) JC ,(NEIBG2(I,JC ),I=1,4)
           WRITE(JDEBUG,4800) JSW,(NEIBG2(I,JSW),I=1,4)
           WRITE(JDEBUG,4900) JS ,(NEIBG2(I,JS ),I=1,4)
           WRITE(JDEBUG,5000) JSE,(NEIBG2(I,JSE),I=1,4)
           WRITE(JDEBUG,5100) JE ,(NEIBG2(I,JE ),I=1,4)
           WRITE(JDEBUG,5200) JNE,(NEIBG2(I,JNE),I=1,4)
           WRITE(JDEBUG,5300) JN ,(NEIBG2(I,JN ),I=1,4)
           WRITE(JDEBUG,5400) JNW,(NEIBG2(I,JNW),I=1,4)
           WRITE(JDEBUG,5500) JW ,(NEIBG2(I,JW ),I=1,4)

           IF (ISTAR1 .NE. 0)
     1          WRITE(JDEBUG,5600) ISTAR1,(NEIBG2(I,ISTAR1),I=1,4)
           IF (ISTAR2 .NE. 0)
     1          WRITE(JDEBUG,5700) ISTAR2,(NEIBG2(I,ISTAR2),I=1,4)
           IF (ISTAR3 .NE. 0)
     1          WRITE(JDEBUG,5800) ISTAR3,(NEIBG2(I,ISTAR3),I=1,4)
           IF (ISTAR4 .NE. 0)
     1          WRITE(JDEBUG,5900) ISTAR4,(NEIBG2(I,ISTAR4),I=1,4)
C
         ENDIF       ! IWRITE
C
C        NOW UPDATE THE NEIGHBOURS OF THE INTERCHANGED CELLS
C
         IF (NEIBG2(3,JSW) .EQ. NLAST1) NEIBG2(3,JSW) = LMSW
         IF (NEIBG2(4,JS ) .EQ. NLAST1) NEIBG2(4,JS ) = LMSW
         IF (NEIBG2(1,JC ) .EQ. NLAST1) NEIBG2(1,JC ) = LMSW
         IF (NEIBG2(2,JW ) .EQ. NLAST1) NEIBG2(2,JW ) = LMSW

         IF (NEIBG2(3,JS ) .EQ. NLAST2) NEIBG2(3,JS ) = LMSE
         IF (NEIBG2(4,JSE) .EQ. NLAST2) NEIBG2(4,JSE) = LMSE
         IF (NEIBG2(1,JE ) .EQ. NLAST2) NEIBG2(1,JE ) = LMSE
         IF (NEIBG2(2,JC ) .EQ. NLAST2) NEIBG2(2,JC ) = LMSE

         IF (NEIBG2(3,JC ) .EQ. NLAST3) NEIBG2(3,JC ) = LMNE
         IF (NEIBG2(4,JE ) .EQ. NLAST3) NEIBG2(4,JE ) = LMNE
         IF (NEIBG2(1,JNE) .EQ. NLAST3) NEIBG2(1,JNE) = LMNE
         IF (NEIBG2(2,JN ) .EQ. NLAST3) NEIBG2(2,JN ) = LMNE

         IF (NEIBG2(3,JW ) .EQ. NLAST4) NEIBG2(3,JW ) = LMNW
         IF (NEIBG2(4,JC ) .EQ. NLAST4) NEIBG2(4,JC ) = LMNW
         IF (NEIBG2(1,JN ) .EQ. NLAST4) NEIBG2(1,JN ) = LMNW
         IF (NEIBG2(2,JNW) .EQ. NLAST4) NEIBG2(2,JNW) = LMNW
C
C        UPDATE STAR EDGE POINTS
```

```
C
      IF (ISTAR1 .NE. 0) THEN
         IF (NEIBG2(1,ISTAR1) .EQ. NLAST1) NEIBG2(1,ISTAR1) = LMSW
         IF (NEIBG2(4,ISTAR1) .EQ. NLAST1) NEIBG2(4,ISTAR1) = LMSW
         IF (NEIBG2(2,ISTAR1) .EQ. NLAST2) NEIBG2(2,ISTAR1) = LMSE
         IF (NEIBG2(3,ISTAR1) .EQ. NLAST2) NEIBG2(3,ISTAR1) = LMSE
      ENDIF
      IF (ISTAR2 .NE. 0) THEN
         IF (NEIBG2(1,ISTAR2) .EQ. NLAST2) NEIBG2(1,ISTAR2) = LMSE
         IF (NEIBG2(2,ISTAR2) .EQ. NLAST2) NEIBG2(2,ISTAR2) = LMSE
         IF (NEIBG2(3,ISTAR2) .EQ. NLAST3) NEIBG2(3,ISTAR2) = LMNE
         IF (NEIBG2(4,ISTAR2) .EQ. NLAST3) NEIBG2(4,ISTAR2) = LMNE
      ENDIF
      IF (ISTAR3 .NE. 0) THEN
         IF (NEIBG2(2,ISTAR3) .EQ. NLAST3) NEIBG2(2,ISTAR3) = LMNE
         IF (NEIBG2(3,ISTAR3) .EQ. NLAST3) NEIBG2(3,ISTAR3) = LMNE
         IF (NEIBG2(4,ISTAR3) .EQ. NLAST4) NEIBG2(4,ISTAR3) = LMNW
         IF (NEIBG2(1,ISTAR3) .EQ. NLAST4) NEIBG2(1,ISTAR3) = LMNW
      ENDIF
      IF (ISTAR4 .NE. 0) THEN
         IF (NEIBG2(3,ISTAR4) .EQ. NLAST4) NEIBG2(3,ISTAR4) = LMNW
         IF (NEIBG2(4,ISTAR4) .EQ. NLAST4) NEIBG2(4,ISTAR4) = LMNW
         IF (NEIBG2(1,ISTAR4) .EQ. NLAST1) NEIBG2(1,ISTAR4) = LMSW
         IF (NEIBG2(2,ISTAR4) .EQ. NLAST1) NEIBG2(2,ISTAR4) = LMSW
      ENDIF
C
C     UPDATE THE OTHER NON-ZERO MIDDLE EDGES
C
      IF (JS1 .NE. 0) THEN
         IF (NEIBG2(3,JS1) .EQ. NLAST1) NEIBG2(3,JS1) = LMSW
         IF (NEIBG2(4,JS1) .EQ. NLAST1) NEIBG2(4,JS1) = LMSW
      ENDIF
      IF (JS2 .NE. 0) THEN
         IF (NEIBG2(3,JS2) .EQ. NLAST2) NEIBG2(3,JS2) = LMSE
         IF (NEIBG2(4,JS2) .EQ. NLAST2) NEIBG2(4,JS2) = LMSE
      ENDIF
C
      IF (JE1 .NE. 0) THEN
         IF (NEIBG2(1,JE1) .EQ. NLAST2) NEIBG2(1,JE1) = LMSE
         IF (NEIBG2(4,JE1) .EQ. NLAST2) NEIBG2(4,JE1) = LMSE
      ENDIF
      IF (JE2 .NE. 0) THEN
         IF (NEIBG2(1,JE2) .EQ. NLAST3) NEIBG2(1,JE2) = LMNE
         IF (NEIBG2(4,JE2) .EQ. NLAST3) NEIBG2(4,JE2) = LMNE
      ENDIF
C
      IF (JN1 .NE. 0) THEN
         IF (NEIBG2(1,JN1) .EQ. NLAST3) NEIBG2(1,JN1) = LMNE
         IF (NEIBG2(2,JN1) .EQ. NLAST3) NEIBG2(2,JN1) = LMNE
      ENDIF
      IF (JN2 .NE. 0) THEN
         IF (NEIBG2(1,JN2) .EQ. NLAST4) NEIBG2(1,JN2) = LMNW
         IF (NEIBG2(2,JN2) .EQ. NLAST4) NEIBG2(2,JN2) = LMNW
      ENDIF
C
      IF (JW1 .NE. 0) THEN
         IF (NEIBG2(2,JW1) .EQ. NLAST4) NEIBG2(2,JW1) = LMNW
```

```
            IF (NEIBG2(3,JW1) .EQ. NLAST4) NEIBG2(3,JW1) = LMNW
          ENDIF
          IF (JW2 .NE. 0) THEN
            IF (NEIBG2(2,JW2) .EQ. NLAST1) NEIBG2(2,JW2) = LMSW
            IF (NEIBG2(3,JW2) .EQ. NLAST1) NEIBG2(3,JW2) = LMSW
          ENDIF
C
C         IF ANY OF THE LAST FOUR CELLS IS DIVIDED, THEN IT IS THE
C         SUPERCELL OF SOME OTHER CELLS NSONJ AND ITS SUPERCELL
C         WILL HAVE TO BE UPDATED

          IF (ICELG2(1,NLAST1) .NE. 0 .OR. ICELG2(1,NLAST2) .NE. 0 .OR.
     1        ICELG2(1,NLAST3) .NE. 0 .OR. ICELG2(1,NLAST4) .NE. 0) THEN
            DO 15 NSONJ = ILVLG2(2,0), NCELG2
              ISUP = ICELG2(10,NSONJ)
              IF (ISUP .GE. NLAST1) THEN
                IF (ISUP .EQ. NLAST1) ICELG2(10,NSONJ) = LMSW
                IF (ISUP .EQ. NLAST2) ICELG2(10,NSONJ) = LMSE
                IF (ISUP .EQ. NLAST3) ICELG2(10,NSONJ) = LMNE
                IF (ISUP .EQ. NLAST4) ICELG2(10,NSONJ) = LMNW
              ENDIF
15          CONTINUE
          ENDIF
C
C         ADJUST ANY BOUNDARY CONDITION POINTERS WHICH POINT TO CELLS
C         JUST INTERCHANGED
C
          IF (IAND(KAUXG2(NLAST1),KLOOOF) .NE. 0 .OR.
     1        IAND(KAUXG2(NLAST2),KLOOOF) .NE. 0 .OR.
     2        IAND(KAUXG2(NLAST3),KLOOOF) .NE. 0 .OR.
     3        IAND(KAUXG2(NLAST4),KLOOOF) .NE. 0     ) THEN
            DO 20 IB = 1, NBNDG2
              IF (IBNDG2(2,IB) .GE. NLAST1) THEN
                 ND1          = IBNDG2(2,IB) - NLAST1
                 IBNDG2(2,IB) = LMSW + ND1
              ENDIF
              IF (IBNDG2(3,IB) .GE. NLAST1) THEN
                 ND1          = IBNDG2(3,IB) - NLAST1
                 IBNDG2(3,IB) = LMSW + ND1
              ENDIF
20          CONTINUE
          ENDIF

C         UPDATE THE ADJACENT CELLS IF NEED BE

          IF (LCSW .GE. NLAST1) LCSW = LMSW + LCSW - NLAST1
          IF (LHSW .GE. NLAST1) LHSW = LMSW + LHSW - NLAST1
          IF (LHSE .GE. NLAST1) LHSE = LMSW + LHSE - NLAST1
          IF (LCSE .GE. NLAST1) LCSE = LMSW + LCSE - NLAST1
          IF (LVSE .GE. NLAST1) LVSE = LMSW + LVSE - NLAST1
          IF (LVNE .GE. NLAST1) LVNE = LMSW + LVNE - NLAST1
          IF (LCNE .GE. NLAST1) LCNE = LMSW + LCNE - NLAST1
          IF (LHNE .GE. NLAST1) LHNE = LMSW + LHNE - NLAST1
          IF (LHNW .GE. NLAST1) LHNW = LMSW + LHNW - NLAST1
          IF (LCNW .GE. NLAST1) LCNW = LMSW + LCNW - NLAST1
          IF (LVNW .GE. NLAST1) LVNW = LMSW + LVNW - NLAST1
          IF (LVSW .GE. NLAST1) LVSW = LMSW + LVSW - NLAST1
```

```
C
C       SET THE NEIGHBOUR-NODE-ARRAY OF THE NEW SUPERCELL
C
        NEIBG2(3,KSW) = LCELL
        NEIBG2(4,KSE) = LCELL
        NEIBG2(1,KNE) = LCELL
        NEIBG2(2,KNW) = LCELL
        NEIBG2(1,KC ) = 0
        NEIBG2(2,KC ) = 0
        NEIBG2(3,KC ) = 0
        NEIBG2(4,KC ) = 0
C
        IF (KS .NE. 0) THEN
          NEIBG2(3,KS)  = LCELL
          NEIBG2(4,KS)  = LCELL
        ELSE
          NEIBG2(1,KSS) = 0
          NEIBG2(2,KSS) = 0
          NEIBG2(3,KSS) = 0
          NEIBG2(4,KSS) = 0
        ENDIF
C
        IF (KE .NE. 0) THEN
          NEIBG2(1,KE)  = LCELL
          NEIBG2(4,KE)  = LCELL
        ELSE
          NEIBG2(1,KEE) = 0
          NEIBG2(2,KEE) = 0
          NEIBG2(3,KEE) = 0
          NEIBG2(4,KEE) = 0
        ENDIF
C
        IF (KN .NE. 0) THEN
          NEIBG2(1,KN)  = LCELL
          NEIBG2(2,KN)  = LCELL
        ELSE
          NEIBG2(1,KNN) = 0
          NEIBG2(2,KNN) = 0
          NEIBG2(3,KNN) = 0
          NEIBG2(4,KNN) = 0
        ENDIF
C
        IF (KW .NE. 0) THEN
          NEIBG2(2,KW)  = LCELL
          NEIBG2(3,KW)  = LCELL
        ELSE
          NEIBG2(1,KWW) = 0
          NEIBG2(2,KWW) = 0
          NEIBG2(3,KWW) = 0
          NEIBG2(4,KWW) = 0
        ENDIF
C
C       ADJUST TOTAL NUMBER OF CELLS
C
        NCELG2 = NCELG2 - NADCEL
C
C       ADJUST THE MAXIMUM LEVEL IF NEED BE
```

```
C
      ILVLG2(3,LEVELO) = ILVLG2(3,LEVELO) - NADCEL
      IF (ILVLG2(3,NLVLG2) .LE. 0) NLVLG2 = NLVLG2 - 1

C     UPDATE THE DIVIDED CELL POINTERS
C
      ICELG2(1,LCELL) = 0
      ICELG2(3,LCELL) = KS
      ICELG2(5,LCELL) = KE
      ICELG2(7,LCELL) = KN
      ICELG2(9,LCELL) = KW

C     RESET EDGE NODE POINTERS OF ALL NEIGHBOURING CELLS

      IF (LS .NE. 0) ICELG2(7,LS) = KS
      IF (LE .NE. 0) ICELG2(9,LE) = KE
      IF (LN .NE. 0) ICELG2(3,LN) = KN
      IF (LW .NE. 0) ICELG2(5,LW) = KW
C
C     SCAN THROUGH ALL BOUNDARY CONDITION POINTERS, LOOKING FOR
C     POINTERS TO THE DIVIDED CELL, SKIP THIS SECTION IF LCELL IS
C     NOT A BOUNDARY CELL

      ITWO = 0
      IMD2 = 0
      ICEN = 0
      IMD1 = 0
      IONE = 0
C
C
C               320        300        280
C                   +-------+-------+
C                   |13D   12C   14E|
C               340 +9   KAUXG2    6+ 260
C                   |11B    3      7|         360 : ERROR
C                   +-------+-------+
C               200        220        240     GO TO STATEMENTS
C
C     BRANCH OUT DEPENDING ON BOUNDARY TYPE
C     (3,S),(6,E),(7,SE),(9,W),(11,SW),(12,N),(13,NW),(14,NE)
C
      IGOTO = IAND (KX,KLOOOF) + 1
      GOTO (370, 360, 360, 220, 360, 360, 260, 240,
     1       360, 340, 360, 200, 300, 320, 280, 360), IGOTO

C
C     SOUTHWESTERN CORNER
C
200   DO 210 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KNW) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KWW) IMD1 = IB
        IF (IBNDG2(1,IB) .EQ. KSW) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KSS) IMD2 = IB
        IF (IBNDG2(1,IB) .EQ. KSE) ITWO = IB
210   CONTINUE
C
C     CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
```

```
C
      NBCPG2(1,1) = IONE
      NBCPG2(1,2) = ITWO
      GO TO 368
C
C     SOUTHERN SIDE
C
220   DO 230 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSW) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KSS) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KSE) ITWO = IB
230   CONTINUE
      GOTO 369
C
C     SOUTHEASTERN CORNER
C
240   DO 250 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSW) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KSS) IMD1 = IB
        IF (IBNDG2(1,IB) .EQ. KSE) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KEE) IMD2 = IB
        IF (IBNDG2(1,IB) .EQ. KNE) ITWO = IB
250   CONTINUE
C
C     CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
C
      NBCPG2(2,1) = IONE
      NBCPG2(2,2) = ITWO
      GO TO 368
C
C     EASTERN SIDE
C
260   DO 270 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSE) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KEE) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KNE) ITWO = IB
270   CONTINUE
      GOTO 369
C
C     NORTHEASTERN CORNER
C
280   DO 290 IB = 1, NBNDG2
        IF (IBNDG2(1,IB) .EQ. KSE) IONE = IB
        IF (IBNDG2(1,IB) .EQ. KEE) IMD1 = IB
        IF (IBNDG2(1,IB) .EQ. KNE) ICEN = IB
        IF (IBNDG2(1,IB) .EQ. KNN) IMD2 = IB
        IF (IBNDG2(1,IB) .EQ. KNW) ITWO = IB
290   CONTINUE
C
C     CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
C
      NBCPG2(3,1) = IONE
      NBCPG2(3,2) = ITWO
      GO TO 368
C
C     NORTHERN SIDE
C
```

```
300     DO 310 IB = 1, NBNDG2
          IF (IBNDG2(1,IB) .EQ. KNE) IONE = IB
          IF (IBNDG2(1,IB) .EQ. KNN) ICEN = IB
          IF (IBNDG2(1,IB) .EQ. KNW) ITWO = IB
310     CONTINUE
        GOTO 369
C
C       NORTHWESTERN CORNER
C
320     DO 330 IB = 1, NBNDG2
          IF (IBNDG2(1,IB) .EQ. KNE) IONE = IB
          IF (IBNDG2(1,IB) .EQ. KNN) IMD1 = IB
          IF (IBNDG2(1,IB) .EQ. KNW) ICEN = IB
          IF (IBNDG2(1,IB) .EQ. KWW) IMD2 = IB
          IF (IBNDG2(1,IB) .EQ. KSW) ITWO = IB
330     CONTINUE
C
C       CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER
C
        NBCPG2(4,1) = IONE
        NBCPG2(4,2) = ITWO
        GO TO 368
C
C       WESTERN SIDE
C
340     DO 350 IB = 1, NBNDG2
          IF (IBNDG2(1,IB) .EQ. KNW) IONE = IB
          IF (IBNDG2(1,IB) .EQ. KWW) ICEN = IB
          IF (IBNDG2(1,IB) .EQ. KSW) ITWO = IB
350     CONTINUE
        GOTO 369
C
C       CHECK THE EDGE CELLS
C       PRINT OUT PARAMETERS FOR BOUNDARY NODES
C
368     IF (IWRITE)  THEN
           WRITE(JDEBUG,6000) IGOTO, KX
           WRITE(JDEBUG,6100) IONE, IMD1, ICEN, IMD2, ITWO
           WRITE(JDEBUG,6200) (IBNDG2(I,IONE),I=1,5)
           WRITE(JDEBUG,6300) (IBNDG2(I,IMD1),I=1,5)
           WRITE(JDEBUG,6400) (IBNDG2(I,ICEN),I=1,5)
           WRITE(JDEBUG,6500) (IBNDG2(I,IMD2),I=1,5)
           WRITE(JDEBUG,6600) (IBNDG2(I,ITWO),I=1,5)
        ENDIF       ! IWRITE

        IF (IONE .EQ. 0 .OR. IMD1 .EQ. 0 .OR. ICEN .EQ. 0
     1                  .OR. IMD2 .EQ. 0 .OR. ITWO .EQ. 0) GOTO 360
C
C       MARK FOR DELETE

        IBNDG2(1,IMD1) = -9
        IBNDG2(1,IMD2) = -9

C       REASSIGN POINTERS

        IBNDG2(3,IONE) = LCELL
        IBNDG2(2,ICEN) = LCELL
```

```
      IBNDG2(2,ITWO) = LCELL

      GOTO 370
C
C     CHECK THE EDGE NODES
C     PRINT OUT PARAMETERS FOR BOUNDARY NODES
C
369   IF (IWRITE)  THEN
         WRITE(JDEBUG,6000) IGOTO, KX
         WRITE(JDEBUG,6100) IONE, IMD1, ICEN, IMD2, ITWO
         WRITE(JDEBUG,6200) (IBNDG2(I,IONE),I=1,5)
         WRITE(JDEBUG,6400) (IBNDG2(I,ICEN),I=1,5)
         WRITE(JDEBUG,6600) (IBNDG2(I,ITWO),I=1,5)
      ENDIF       ! IWRITE

C     IF (IONE .EQ. 0 .OR. ICEN .EQ. 0 .OR. ITWO .EQ. 0) GOTO 360
      IF (ICEN .EQ. 0) GOTO 360
C
C     MARK FOR DELETE

      IBNDG2(1,ICEN) = -9

C     REASSIGN POINTERS

      IBNDG2(3,IONE) = LCELL
      IBNDG2(2,ITWO) = LCELL

      GOTO 370
C
C     ERROR IN BOUNDARY CELL POINTERS

360   ZER1 = LCELL
      ZER2 = IGOTO
      CALL ERRORM (17,'G2CLPO','LCELL ',ZER1,'IGOTO ',ZER2,JPRINT,
     1     'ERROR IN BOUNDARY NODE CALCULATION')

370   CONTINUE
C
C     -----------
C     DEBUG PRINT
C     -----------
C
C     PRINT OUT PARAMETERS AFTER CELL MERGER
C
      IF (IWRITE)  THEN
         WRITE(JDEBUG,6700)
C
C        GENERAL INFORMATION
C
         WRITE(JDEBUG,1400) NNODG2, NCELG2, NBNDG2, LEVELG
C
C        POINTERS OF MAIN CELL LCELL
C
         WRITE(JDEBUG,1500) LCELL, KC , KSW, KS , KSE, KE,
     1                             KNE, KN , KNW, KW , KX
         WRITE(JDEBUG,1600) (ICELG2(I,LCELL),I=1,10),KAUXG2(LCELL)
C
```

```
C          REASSIGNED CELLS
C
           WRITE(JDEBUG,6800) LMSW, LMSE, LMNE, LMNW
           WRITE(JDEBUG,1800) (ICELG2(I,LMSW),I=1,10),KAUXG2(LMSW)
           WRITE(JDEBUG,1900) (ICELG2(I,LMSE),I=1,10),KAUXG2(LMSE)
           WRITE(JDEBUG,2000) (ICELG2(I,LMNE),I=1,10),KAUXG2(LMNE)
           WRITE(JDEBUG,2100) (ICELG2(I,LMNW),I=1,10),KAUXG2(LMNW)
C
C          NEIGHBOUR CELLS AND THEIR POINTERS
C
           WRITE(JDEBUG,2200) LVSW, LCSW, LHSW, LHSE, LCSE, LVSE,
     1                        LVNE, LCNE, LHNE, LHNW, LCNW, LVNW

           IF (LVSW .NE. 0) THEN
              WRITE(JDEBUG,2300) (ICELG2(I,LVSW),I=1,10),KAUXG2(LVSW)
           ENDIF
           IF (LCSW .NE. 0) THEN
              WRITE(JDEBUG,2400) (ICELG2(I,LCSW),I=1,10),KAUXG2(LCSW)
           ENDIF
           IF (LHSW .NE. 0) THEN
              WRITE(JDEBUG,2500) (ICELG2(I,LHSW),I=1,10),KAUXG2(LHSW)
           ENDIF
C
           IF (LHSE .NE. 0) THEN
              WRITE(JDEBUG,2600) (ICELG2(I,LHSE),I=1,10),KAUXG2(LHSE)
           ENDIF
           IF (LCSE .NE. 0) THEN
              WRITE(JDEBUG,2700) (ICELG2(I,LCSE),I=1,10),KAUXG2(LCSE)
           ENDIF
           IF (LVSE .NE. 0) THEN
              WRITE(JDEBUG,2800) (ICELG2(I,LVSE),I=1,10),KAUXG2(LVSE)
           ENDIF
C
           IF (LVNE .NE. 0) THEN
              WRITE(JDEBUG,2900) (ICELG2(I,LVNE),I=1,10),KAUXG2(LVNE)
           ENDIF
           IF (LCNE .NE. 0) THEN
              WRITE(JDEBUG,3000) (ICELG2(I,LCNE),I=1,10),KAUXG2(LCNE)
           ENDIF
           IF (LHNE .NE. 0) THEN
              WRITE(JDEBUG,3100) (ICELG2(I,LHNE),I=1,10),KAUXG2(LHNE)
           ENDIF
C
           IF (LHNW .NE. 0) THEN
              WRITE(JDEBUG,3200) (ICELG2(I,LHNW),I=1,10),KAUXG2(LHNW)
           ENDIF
           IF (LCNW .NE. 0) THEN
              WRITE(JDEBUG,3300) (ICELG2(I,LCNW),I=1,10),KAUXG2(LCNW)
           ENDIF
           IF (LVNW .NE. 0) THEN
              WRITE(JDEBUG,3400) (ICELG2(I,LVNW),I=1,10),KAUXG2(LVNW)
           ENDIF
C
C          NEIGHBOURING CELLS OF ALL NODES OF LCELL
C
           WRITE(JDEBUG,3500) (NEIBG2(I,KC ),I=1,4)
           WRITE(JDEBUG,3600) (NEIBG2(I,KSW),I=1,4)
```

```
            WRITE(JDEBUG,3800) (NEIBG2(I,KSE),I=1,4)
            WRITE(JDEBUG,4000) (NEIBG2(I,KNE),I=1,4)
            WRITE(JDEBUG,4200) (NEIBG2(I,KNW),I=1,4)
            WRITE(JDEBUG,3700) (NEIBG2(I,KS ),I=1,4)
            WRITE(JDEBUG,3900) (NEIBG2(I,KE ),I=1,4)
            WRITE(JDEBUG,4100) (NEIBG2(I,KN ),I=1,4)
            WRITE(JDEBUG,4300) (NEIBG2(I,KW ),I=1,4)
C
C           BOUNDARY NODES
C
C       PRINT OUT PARAMETERS FOR BOUNDARY NODES
C
            IF (IGOTO .NE. 0) THEN
               WRITE(JDEBUG,6100) IONE, IMD1, ICEN, IMD2, ITWO
               WRITE(JDEBUG,6100) IONE, ICEN, ITWO
               WRITE(JDEBUG,6200) (IBNDG2(I,IONE),I=1,5)
               WRITE(JDEBUG,6400) (IBNDG2(I,ICEN),I=1,5)
               WRITE(JDEBUG,6600) (IBNDG2(I,ITWO),I=1,5)
               IF (IMD1 .NE. 0) THEN
                 WRITE(JDEBUG,6300) (IBNDG2(I,IMD1),I=1,5)
                 WRITE(JDEBUG,6500) (IBNDG2(I,IMD2),I=1,5)
               ENDIF
            ENDIF
         ENDIF       ! IWRITE
C
C       CHECK IF THE CELL HAS FUEL INJECTED TO IT
C
         IF (IAND(KX,KL1000) .EQ. 0) RETURN

         KUMDH2 = 0

         DO 380 IB = 1, NUMDH2
           IF (NODEH2(IB) .EQ. KWW) THEN
              IBHERE = IB
              GOTO 390
           ENDIF
380      CONTINUE

390      DO 400 IB = IBHERE, NUMDH2-1
           NODEH2(IB) = NODEH2(IB+1)
400      CONTINUE

         NUMDH2 = NUMDH2 - 1
C
C        ----------------
C        FORMAT STATEMENTS
C        ----------------
C
1000     FORMAT(//10X,'----------------------' )
1100     FORMAT(  10X,'DEBUG PRINT FROM G2CLPO' )
1200     FORMAT(  10X,'----------------------'/)
1300     FORMAT(/10X,'***** INFORMATION BEFORE COLLAPSE *****'/)
1400     FORMAT(5X,'NNODG2 =',I7,5X,'NCELG2 =',I7,5X,'NBNDG2 =',I7,
     1         5X,'LEVEL  =',I7                                  )
1500     FORMAT(5X,'CELL POINTERS FOR LCELL =',I6,/5X,'KC  =',I6,5X,
     1    'KSW =',I6,5X,'KS  =',I6,5X,'KSE =',I6,5X,'KE  =',I6/5X,
     2    'KNE =',I6,5X,'KN  =',I6,5X,'KNW =',I6,5X,'KW  =',I6,5X,
```

```
     3    'KX  =',Z7                                                    )
1600    FORMAT(5X,'LCELL POINTERS',5X,10I6,Z10)
1700    FORMAT(5X,'MIDDLE CELLS ARE :'/
     1         5X,'LMSW=',I5,5X,'LMSE=',I5,5X,'LMNE=',I5,
     2         5X,'LMNW=',I5                                          )
1800    FORMAT(5X,'LMSW  POINTERS',5X,10I6,Z10)
1900    FORMAT(5X,'LMSE  POINTERS',5X,10I6,Z10)
2000    FORMAT(5X,'LMNE  POINTERS',5X,10I6,Z10)
2100    FORMAT(5X,'LMNW  POINTERS',5X,10I6,Z10)
2200    FORMAT(5X,'CELLS NEIGHBOURING LCELL :'/
     1         5X,'LVSW=',I5,5X,'LCSW=',I5,5X,'LHSW=',I5,
     2         5X,'LHSE=',I5,5X,'LCSE=',I5,5X,'LVSE=',I5/
     3         5X,'LVNE=',I5,5X,'LCNE=',I5,5X,'LHNE=',I5,
     4         5X,'LHNW=',I5,5X,'LCNW=',I5,5X,'LVNW=',I5   )
2300    FORMAT(5X,'LVSW  POINTERS',5X,10I6,Z10)
2400    FORMAT(5X,'LCSW  POINTERS',5X,10I6,Z10)
2500    FORMAT(5X,'LHSW  POINTERS',5X,10I6,Z10)
2600    FORMAT(5X,'LHSE  POINTERS',5X,10I6,Z10)
2700    FORMAT(5X,'LCSE  POINTERS',5X,10I6,Z10)
2800    FORMAT(5X,'LVSE  POINTERS',5X,10I6,Z10)
2900    FORMAT(5X,'LVNE  POINTERS',5X,10I6,Z10)
3000    FORMAT(5X,'LCNE  POINTERS',5X,10I6,Z10)
3100    FORMAT(5X,'LHNE  POINTERS',5X,10I6,Z10)
3200    FORMAT(5X,'LHNW  POINTERS',5X,10I6,Z10)
3300    FORMAT(5X,'LCNW  POINTERS',5X,10I6,Z10)
3400    FORMAT(5X,'LVNW  POINTERS',5X,10I6,Z10)
3500    FORMAT(5X,'NEIGHBOUR CELLS OF KC  :',4I7)
3600    FORMAT(5X,'NEIGHBOUR CELLS OF KSW :',4I7)
3700    FORMAT(5X,'NEIGHBOUR CELLS OF KS  :',4I7)
3800    FORMAT(5X,'NEIGHBOUR CELLS OF KSE :',4I7)
3900    FORMAT(5X,'NEIGHBOUR CELLS OF KE  :',4I7)
4000    FORMAT(5X,'NEIGHBOUR CELLS OF KNE :',4I7)
4100    FORMAT(5X,'NEIGHBOUR CELLS OF KN  :',4I7)
4200    FORMAT(5X,'NEIGHBOUR CELLS OF KNW :',4I7)
4300    FORMAT(5X,'NEIGHBOUR CELLS OF KW  :',4I7)
4400    FORMAT(5X,'LAST FOUR CELLS ARE =',/5X,'NLAST1=',I6,5X,
     1    'NLAST2 =',I6,5X,'NLAST3 =',I6,5X,'NLAST4 =',I6)

4500    FORMAT(5X,'NODE POINTERS OF LAST FOUR CELLS =',/5X,'JC  =',
     1    I6,5X,'JSW =',I6,5X,'JS  =',I6,5X,'JSE =',I6,5X,'JE  =',I6/
     2       5X,'JNE =',I6,5X,'JN  =',I6,5X,'JNW =',I6,5X,'JW  =',I6/
     3       5X,'ISTAR1=',I6,5X,'ISTAR2=',I6,5X,'ISTAR3=',I6,
     4       5X,'ISTAR4=',I6)
4600    FORMAT(5X,'CELL POINTERS OF NLAST',I1,5X,10I6,Z10)
4700    FORMAT(5X,'NEIGHBOUR CELLS OF JC  :',5I7)
4800    FORMAT(5X,'NEIGHBOUR CELLS OF JSW :',5I7)
4900    FORMAT(5X,'NEIGHBOUR CELLS OF JS  :',5I7)
5000    FORMAT(5X,'NEIGHBOUR CELLS OF JSE :',5I7)
5100    FORMAT(5X,'NEIGHBOUR CELLS OF JE  :',5I7)
5200    FORMAT(5X,'NEIGHBOUR CELLS OF JNE :',5I7)
5300    FORMAT(5X,'NEIGHBOUR CELLS OF JN  :',5I7)
5400    FORMAT(5X,'NEIGHBOUR CELLS OF JNW :',5I7)
5500    FORMAT(5X,'NEIGHBOUR CELLS OF JW  :',5I7)
5600    FORMAT(5X,'NEIGHBOUR CELLS OF ISTAR1  :',5I7)
5700    FORMAT(5X,'NEIGHBOUR CELLS OF ISTAR1  :',5I7)
5800    FORMAT(5X,'NEIGHBOUR CELLS OF ISTAR1  :',5I7)
5900    FORMAT(5X,'NEIGHBOUR CELLS OF ISTAR1  :',5I7)
```

```
6000    FORMAT(5X,'IGOTO =',I3,2X,'KX =',Z10)
6100    FORMAT(5X,'BOUNDARY NODE INFORMATION : '/
     1      - 5X,'IONE =',I6,5X,'IMD1 =',I6,5X,'ICEN =',I6,
     2        5X,'IMD2 =',I6,5X,'ITWO =',I6)
6200    FORMAT(5X,'B. POINTERS OF IONE :',5I6)
6300    FORMAT(5X,'B. POINTERS OF IMD1 :',5I6)
6400    FORMAT(5X,'B. POINTERS OF ICEN :',5I6)
6500    FORMAT(5X,'B. POINTERS OF IMD2 :',5I6)
6600    FORMAT(5X,'B. POINTERS OF ITWO :',5I6)
6700    FORMAT(//10X,'***** INFORMATION AFTER  COLLAPSE *****'/)
6800    FORMAT(5X,'REASSIGNED MIDDLE CELLS ARE :'/
     1        5X,'LMSW=',I5,5X,'LMSE=',I5,5X,'LMNE=',I5,
     2        5X,'LMNW=',I5                              )

        RETURN
        END
```

G2DIVU

```
        SUBROUTINE G2DIVO (LCELL, IWARN)
C                 G2DIVU

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'H2COMN.INC'
        INCLUDE 'HEXCOD.INC'
        DIMENSION IBNODE(5)
C
C***********************************************************************
C
C       THIS SUBROUTINE DIVIDES CELL 'LCELL' INTO FOUR SMALLER CELLS
C       AND PERFORMS ALL NECESSARY POINTER SYSTEM REALIGNMENTS
C
C
C       SPECIAL EXPLANATION OF AUXILIARY CELL POINTERS
C
C         KAUXG2(LCELL) HAS THE HEXIDECIMAL FORM:
C
C                     'X  X  X  X  X  X  X  X '
C                       8  7  6  5  4  3  2  1
C
C          WHERE:
C              X    INDICATES THAT CELL IS A BOUNDARY CELL
C               1
C
C              X    INDICATES THAT THE CELL WAS RECENTLY DIVIDED
C               2       AND HENCE MUST NOT BE COLLAPSED
C
C              X    INDICATES THE BOUNDARY INTERPOLATION FUNCTION TYPE
C               3
C
C              X    INDICATES SPECIAL CELLS (E.G., FUEL INJECTION CELLS)
```

```
C             4
C
C          - X    INDICATES THE SPATIAL LEVEL OF THE  CELL ( <= MLVLG2)
C             5
C
C          EXPLANATION OF HISTORY POINTER OF THE CELLS (BYTE 2)
C
C          THE CELL WHICH IS DIVIDED CAN ONLY BE COLLAPSED AFTER THREE
C          GENERATIONS LATER, NOTE THAT INCHIS=48=2**5+2**4 (I.E., THE
C          SECOND BYTE IS SET EQUAL TO 3).  AFTER EACH ADAPTATION CYCLE
C          (I.E., A CALL TO A2MTHO) THIS POINTER IS REDUCED BY ONE UNTIL
C          IT BECOMES ZERO.
C
C          EXPLANATION OF SPATIAL LEVEL POINTER OF THE CELLS (BYTE 5)
C
C          K5LEVG : 5TH BYTE OF THE GIVEN CELL LCELL
C          K5LEVN : 5TH BYTE OF THE NEW CELLS
C          LEVELG : LEVEL OF THE GIVEN CELL LCELL
C          LEVELN : LEVEL OF THE NEW CELLS (TO BE DIVIDED, LEVELG+1)

C*********************************************************************
C
        INCHIS = 48
C
C       --------------
C       OVERFLOW CHECK
C       --------------
C
C       CHECK FOR OVERFLOW IN NODE ARRAYS
C
        IF(NNODG2+5 .GT. MNODG2) THEN
          ZER1 = MNODG2
          ZER2 = NNODG2
          CALL WARNIN (6,'G2DIVO','MNODG2',ZER1,'NNODG2',ZER2,JPRINT,
     1      'NUMBER OF NODES EXCEEDS ITS LIMIT')
          IWARN = 6
          RETURN
        ENDIF

C       CHECK FOR OVERFLOW IN CELL LIMIT

        IF (NCELG2+4 .GT. MCELG2) THEN
          ZER1 = MCELG2
          ZER2 = NCELG2
          CALL WARNIN (7,'G2DIVO','MCELG2',ZER1,'NCELG2',ZER2,JPRINT,
     1      'NUMBER OF CELLS EXCEEDS ITS LIMIT')
          IWARN = 7
          RETURN
        ENDIF

C       CHECK FOR OVERFLOW IN BOUNDARY CONDITION ARRAY

        IF (NBNDG2+2 .GT. MBNDG2) THEN
          ZER1 = MBNDG2
          ZER2 = NBNDG2
          CALL WARNIN (8,'G2DIVO','MBNDG2',ZER1,'NBNDG2',ZER2,JPRINT,
     1      'NUMBER OF BOUNDARY NODES EXCEEDS ITS LIMIT')
```

```
          IWARN = 8
          RETURN
        ENDIF

C       FIND THE LEVEL LEVELG OF THE GIVEN CELL
        KX     = KAUXG2(LCELL)
        K5LEVG = IAND(KX,KU000F)
        LEVELG = ISHFT(K5LEVG,-16)

C
C       --------------
C       POINTER SAVING
C       --------------
C
C       SAVE REST OF CELL POINTERS

        KSW    = ICELG2(2,LCELL)
        KS     = ICELG2(3,LCELL)
        KSE    = ICELG2(4,LCELL)
        KE     = ICELG2(5,LCELL)
        KNE    = ICELG2(6,LCELL)
        KN     = ICELG2(7,LCELL)
        KNW    = ICELG2(8,LCELL)
        KW     = ICELG2(9,LCELL)
        K5LEVN = K5LEVG + 2**16
        LEVELN = LEVELG + 1
        MAXLEV = MAX(NLVLG2, LEVELN)

        IF (LEVELN .GT. MALVG2) RETURN
C
C       ----------------------
C       NEIGHBOUR DETERMINATION
C       ----------------------
C
C       FIND CELLS WHICH BOUND DIVIDED CELL
C
C         |------------|--------------|------------|
C         |            |              |            |  K FOR NODE
C         |      + + + + + + + + + + + + + + +      |  L FOR CELL
C         |      +LCNW | LHNW  + LHNE  |LCNE +      |
C         |      +     |       +       |     +      |
C         |------+-----+-------+-------+-----+------|
C         |      +     |KNW   KN    KNE|     +      | M:CENTER (MIDDLE)
C         |      +LVNW | LMNW     LMNE |LVNE +      | C:CORNER (ADJACENT)
C         |      + + + +KW   LCELL   KE+ + + +      | H:HORIZONTAL (ADJAC)
C         |      +LVSW | LMSW     LMSE |LVSE +      | V:VERTICAL (ADJACENT)
C         |      +     |KSW   KS    KSE|     +      |
C         |------+-----+-------+-------+-----+------|
C         |      +     |       +       |     +      |
C         |      +LCSW | LHSW  + LHSE  |LCSE +      |
C         |      + + + + + + + + + + + + + + +      |
C         |            |              |            |
C         |------------|--------------|------------|
C
          LVSW   = NEIBG2(4,KSW)
          LCSW   = NEIBG2(1,KSW)
          LHSW   = NEIBG2(2,KSW)
```

```
      LHSE   = NEIBG2(1,KSE)
      LCSE-  = NEIBG2(2,KSE)
      LVSE-  = NEIBG2(3,KSE)
      LVNE   = NEIBG2(2,KNE)
      LCNE   = NEIBG2(3,KNE)
      LHNE   = NEIBG2(4,KNE)
      LHNW   = NEIBG2(3,KNW)
      LCNW   = NEIBG2(4,KNW)
      LVNW   = NEIBG2(1,KNW)
C
C     ------------------------
C     LEVEL DIFFERENTIAL CHECK
C     ------------------------
C
C     IF THE COMPONENT CELLS ARE JUST INSIDE EMBEDDED REGION THEN
C     THEY CAN NOT BE DIVIDED; THIS WILL BE SO IF THE LEVELS OF
C     THE NEIGHBOURHOOD CELLS DIFFER BY MORE THAN ONE
C     FIRST DO THE CORNER CELLS

      IF (LCSW .NE. 0) THEN
        K5LCOR = IAND(KAUXG2(LCSW),KU000F)
        LEVELC = ISHFT(K5LCOR,-16)
        IDFL   = LEVELC-LEVELG
        IF (IDFL .LT. 0) RETURN
      ENDIF
C
      IF (LCSE .NE. 0) THEN
        K5LCOR = IAND(KAUXG2(LCSE),KU000F)
        LEVELC = ISHFT(K5LCOR,-16)
        IDFL   = LEVELC-LEVELG
        IF (IDFL .LT. 0) RETURN
      ENDIF
C
      IF (LCNE .NE. 0) THEN
        K5LCOR = IAND(KAUXG2(LCNE),KU000F)
        LEVELC = ISHFT(K5LCOR,-16)
        IDFL   = LEVELC-LEVELG
        IF (IDFL .LT. 0) RETURN
      ENDIF
C
      IF (LCNW .NE. 0) THEN
        K5LCOR = IAND(KAUXG2(LCNW),KU000F)
        LEVELC = ISHFT(K5LCOR,-16)
        IDFL   = LEVELC-LEVELG
        IF (IDFL .LT. 0) RETURN
      ENDIF
C
C     NOW DO EDGE CELLS
C
      IF (LHSW .NE. 0) THEN
        K5LEDG = IAND(KAUXG2(LHSW),KU000F)
        LEVELC = ISHFT(K5LEDG,-16)
        IDFL   = LEVELC-LEVELG
        IF (IDFL .LT. 0) RETURN
      ENDIF
C
      IF (LVSE .NE. 0) THEN
```

```
          K5LEDG = IAND(KAUXG2(LVSE),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0) RETURN
        ENDIF
C
        IF (LHNE .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LHNE),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0) RETURN
        ENDIF
C
        IF (LVNW .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LVNW),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0) RETURN
        ENDIF
C
C       ----------------
C       DIVISION PROCESS
C       ----------------
C
C       INITIATE THE PROCESS OF CELL DIVISION
C       UPDATE THE OVERALL MAXIMUM LEVEL POINTER
C
        NLVLG2 = MAXLEV

C       UPDATE THE NUMBER OF CELLS AT THE NEW LEVEL

        ILVLG2(3,LEVELN) = ILVLG2(3,LEVELN) + 4

C       CREATE NODE AT CENTER OF CELL

        NNODG2 = NNODG2 + 1
        KC     = NNODG2

C       COMPUTE THE GEOMETRIC QUANTITIES AT THE NEW CENTER NODE

        GEOMG2(1,KC) = 0.25*( GEOMG2(1,KSW) + GEOMG2(1,KSE) +
     1                        GEOMG2(1,KNE) + GEOMG2(1,KNW) )
        GEOMG2(2,KC) = 0.25*( GEOMG2(2,KSW) + GEOMG2(2,KSE) +
     1                        GEOMG2(2,KNE) + GEOMG2(2,KNW) )

C       LINEAR INTERPOLATION FOR DEPENDENT VARIABLES

        DO 10 J = 1, NEQNFL
          DPENG2(J,KC) = 0.25*( DPENG2(J,KSW) + DPENG2(J,KSE) +
     1                          DPENG2(J,KNE) + DPENG2(J,KNW) )
10      CONTINUE
          PRESG2(KC) = 0.25*( PRESG2(KSW) + PRESG2(KSE) +
     1                        PRESG2(KNE) + PRESG2(KNW) )
          TEMPG2(KC) = 0.25*( TEMPG2(KSW) + TEMPG2(KSE) +
     1                        TEMPG2(KNE) + TEMPG2(KNW) )

C       DOES SOUTHERN NODE ALREADY EXIST; IF NOT CREATE IT
```

```
      IF (KS .EQ. 0) THEN
        NNODG2       = NNODG2 + 1
        KS           = NNODG2
        GEOMG2(1,KS) = 0.50*( GEOMG2(1,KSW) + GEOMG2(1,KSE) )
        GEOMG2(2,KS) = 0.50*( GEOMG2(2,KSW) + GEOMG2(2,KSE) )
        IF (LHSW .NE. 0 .AND. LHSE .NE. 0) THEN
          NEIBG2(1,KS) = LHSW
          NEIBG2(2,KS) = LHSE
        ELSE
          NEIBG2(1,KS) = 0
          NEIBG2(2,KS) = 0
        ENDIF
        DO 20 J = 1, NEQNFL
          DPENG2(J,KS) = 0.50*( DPENG2(J,KSW) + DPENG2(J,KSE) )
20      CONTINUE
        PRESG2(KS) = 0.50*( PRESG2(KSW) + PRESG2(KSE) )
        TEMPG2(KS) = 0.50*( TEMPG2(KSW) + TEMPG2(KSE) )
      ENDIF

C     DOES EASTERN  NODE ALREADY EXIST; IF NOT CREATE IT

      IF (KE .EQ. 0) THEN
        NNODG2       = NNODG2 + 1
        KE           = NNODG2
        GEOMG2(1,KE) = 0.50*( GEOMG2(1,KNE) + GEOMG2(1,KSE) )
        GEOMG2(2,KE) = 0.50*( GEOMG2(2,KNE) + GEOMG2(2,KSE) )
        IF (LVSE .NE. 0 .AND. LVNE .NE. 0) THEN
          NEIBG2(2,KE) = LVSE
          NEIBG2(3,KE) = LVNE
        ELSE
          NEIBG2(2,KE) = 0
          NEIBG2(3,KE) = 0
        ENDIF
        DO 30 J = 1, NEQNFL
          DPENG2(J,KE) = 0.50*( DPENG2(J,KNE) + DPENG2(J,KSE) )
30      CONTINUE
        PRESG2(KE) = 0.50*( PRESG2(KNE) + PRESG2(KSE) )
        TEMPG2(KE) = 0.50*( TEMPG2(KNE) + TEMPG2(KSE) )
      ENDIF

C     DOES NORTHERN NODE ALREADY EXIST; IF NOT CREATE IT

      IF (KN .EQ. 0) THEN
        NNODG2       = NNODG2 + 1
        KN           = NNODG2
        GEOMG2(1,KN) = 0.50*( GEOMG2(1,KNE) + GEOMG2(1,KNW) )
        GEOMG2(2,KN) = 0.50*( GEOMG2(2,KNE) + GEOMG2(2,KNW) )
        IF (LHNW .NE. 0 .AND. LHNE .NE. 0) THEN
          NEIBG2(3,KN) = LHNE
          NEIBG2(4,KN) = LHNW
        ELSE
          NEIBG2(3,KN) = 0
          NEIBG2(4,KN) = 0
        ENDIF
        DO 40 J = 1, NEQNFL
          DPENG2(J,KN) = 0.50*( DPENG2(J,KNE) + DPENG2(J,KNW) )
```

```
40        CONTINUE
          PRESG2(KN) = 0.50*( PRESG2(KNE) + PRESG2(KNW) )
          TEMPG2(KN) = 0.50*( TEMPG2(KNE) + TEMPG2(KNW) )
        ENDIF

C       DOES WESTERN NODE ALREADY EXIST; IF NOT CREATE IT

        IF (KW .EQ. 0) THEN
          NNODG2       = NNODG2 + 1
          KW           = NNODG2
          GEOMG2(1,KW) = 0.50*( GEOMG2(1,KNW) + GEOMG2(1,KSW) )
          GEOMG2(2,KW) = 0.50*( GEOMG2(2,KNW) + GEOMG2(2,KSW) )
          IF (LVSW .NE. 0 .AND. LVNW .NE. 0) THEN
            NEIBG2(1,KW) = LVSW
            NEIBG2(4,KW) = LVNW
          ELSE
            NEIBG2(1,KW) = 0
            NEIBG2(4,KW) = 0
          ENDIF
          DO 50 J = 1, NEQNFL
            DPENG2(J,KW) = 0.50*( DPENG2(J,KNW) + DPENG2(J,KSW) )
50        CONTINUE
          PRESG2(KW) = 0.50*( PRESG2(KNW) + PRESG2(KSW) )
          TEMPG2(KW) = 0.50*( TEMPG2(KNW) + TEMPG2(KSW) )
        ENDIF

C       UPDATE THE DIVIDED CELL -- NEW NODE

        ICELG2(1,LCELL) = KC
        ICELG2(3,LCELL) = KS
        ICELG2(5,LCELL) = KE
        ICELG2(7,LCELL) = KN
        ICELG2(9,LCELL) = KW

C       CREATE THE NEW CELLS

        LMSW   = NCELG2 + 1
        LMSE   = LMSW   + 1
        LMNE   = LMSE   + 1
        LMNW   = LMNE   + 1
        NCELG2 = NCELG2 + 4

        ICELG2( 1,LMSW) = 0
        ICELG2( 2,LMSW) = KSW
        ICELG2( 3,LMSW) = 0
        ICELG2( 4,LMSW) = KS
        ICELG2( 5,LMSW) = 0
        ICELG2( 6,LMSW) = KC
        ICELG2( 7,LMSW) = 0
        ICELG2( 8,LMSW) = KW
        ICELG2( 9,LMSW) = 0
        ICELG2(10,LMSW) = LCELL
        KAUXG2(   LMSW) = K5LEVN + INCHIS

        ICELG2( 1,LMSE) = 0
        ICELG2( 2,LMSE) = KS
        ICELG2( 3,LMSE) = 0
```

```
      ICELG2( 4,LMSE) = KSE
      ICELG2( 5,LMSE) = 0
      ICELG2( 6,LMSE) = KE
      ICELG2( 7,LMSE) = 0
      ICELG2( 8,LMSE) = KC
      ICELG2( 9,LMSE) = 0
      ICELG2(10,LMSE) = LCELL
      KAUXG2(   LMSE) = K5LEVN + INCHIS

      ICELG2( 1,LMNE) = 0
      ICELG2( 2,LMNE) = KC
      ICELG2( 3,LMNE) = 0
      ICELG2( 4,LMNE) = KE
      ICELG2( 5,LMNE) = 0
      ICELG2( 6,LMNE) = KNE
      ICELG2( 7,LMNE) = 0
      ICELG2( 8,LMNE) = KN
      ICELG2( 9,LMNE) = 0
      ICELG2(10,LMNE) = LCELL
      KAUXG2(   LMNE) = K5LEVN + INCHIS

      ICELG2( 1,LMNW) = 0
      ICELG2( 2,LMNW) = KW
      ICELG2( 3,LMNW) = 0
      ICELG2( 4,LMNW) = KC
      ICELG2( 5,LMNW) = 0
      ICELG2( 6,LMNW) = KN
      ICELG2( 7,LMNW) = 0
      ICELG2( 8,LMNW) = KNW
      ICELG2( 9,LMNW) = 0
      ICELG2(10,LMNW) = LCELL
      KAUXG2(   LMNW) = K5LEVN + INCHIS

C     SET EDGE NODE POINTERS OF ALL NEIGHBOURING CELLS

      IF (LHSW.NE.0 .AND. LHSW.EQ.LHSE) ICELG2( 7,LHSW) = KS
      IF (LVSE.NE.0 .AND. LVSE.EQ.LVNE) ICELG2( 9,LVSE) = KE
      IF (LHNE.NE.0 .AND. LHNE.EQ.LHNW) ICELG2( 3,LHNE) = KN
      IF (LVNW.NE.0 .AND. LVNW.EQ.LVSW) ICELG2( 5,LVNW) = KW

C     UPDATE NEIGHBOUR-NODE-ARRAY

      NEIBG2(1,KC ) = LMSW
      NEIBG2(2,KC ) = LMSE
      NEIBG2(3,KC ) = LMNE
      NEIBG2(4,KC ) = LMNW
      NEIBG2(3,KSW) = LMSW
      NEIBG2(3,KS ) = LMSE
      NEIBG2(4,KS ) = LMSW
      NEIBG2(4,KSE) = LMSE
      NEIBG2(1,KE ) = LMSE
      NEIBG2(4,KE ) = LMNE
      NEIBG2(1,KNE) = LMNE
      NEIBG2(1,KN ) = LMNW
      NEIBG2(2,KN ) = LMNE
      NEIBG2(2,KNW) = LMNW
      NEIBG2(2,KW ) = LMSW
```

```
      NEIBG2(3,KW ) = LMNW
C
C     ----------------------
C     BOUNDARY NODE POINTERS
C     ----------------------
C
C     SKIP NEXT SECTION IF LCELL IS NOT A BOUNDARY CELL
C
      IGOTO = IAND(KX,KLOOOF)
      IF(IGOTO .EQ. 0) GO TO 290
C
C     SCAN THROUGH ALL BOUNDARY CONDITION POINTERS, LOOKING FOR POINTERS
C     TO THE DIVIDED CELL
C
      IONE = 0
      ICOR = 0
      ITWO = 0
      DO 60 IBND = 1, NBNDG2
         IF (IBNDG2(3,IBND) .EQ. LCELL      ) IONE = IBND
         IF (IBNDG2(2,IBND) .EQ. LCELL .AND.
     1       IBNDG2(3,IBND) .EQ. 0          ) ICOR = IBND
         IF (IBNDG2(2,IBND) .EQ. LCELL .AND.
     1       IBNDG2(3,IBND) .NE. 0          ) ITWO = IBND
60    CONTINUE
C
C     SPECIAL INJECTOR CASE
      IF (ITWO .EQ. 0 .AND. ICOR .NE. 0) ITWO = ICOR
C
C
C               260        250        240
C                   +-------+-------+
C                   |13D   12C   14E|
C               270 +9   KAUXG2    6+ 230
C                   |11B    3      7|         280 : ERROR
C                   +-------+-------+
C               200        210        220     GO TO STATEMENTS
C
C
C     BRANCH OUT DEPENDING ON BOUNDARY TYPE
C
      GO TO (280,280,280,210,280,280,230,220,
     1       280,270,280,200,250,260,240,280), (IGOTO + 1)
      GO TO 280

C     DIVIDED CELL WAS AT SOUTHWESTERN CORNER

200   IBNDG2(2,ICOR)   = LMSW
      IBNDG2(2,ITWO)   = LMSE
      IBNDG2(3,IONE)   = LMNW

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KS
      IBNDG2(2,NBNDG2) = LMSW
      IBNDG2(3,NBNDG2) = LMSE
      IBNDG2(4,NBNDG2) = 3
      IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
      NBCPG2(1,2)      = NBNDG2
```

```
      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KW
      IBNDG2(2,NBNDG2) = LMNW
      IBNDG2(3,NBNDG2) = LMSW
      IBNDG2(4,NBNDG2) = 9
      IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
      NBCPG2(1,1)      = NBNDG2

      KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL0009)
      KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL000B)
      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0003)

      GO TO 290

C     DIVIDED CELL WAS ALONG SOUTHERN EDGE

210   IBNDG2(3,IONE)   = LMSW
      IBNDG2(2,ITWO)   = LMSE

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KS
      IBNDG2(2,NBNDG2) = LMSW
      IBNDG2(3,NBNDG2) = LMSE
      IBNDG2(4,NBNDG2) = 3

      I5ONE = IBNDG2(5,IONE)
      I5TWO = I5ONE
      IF (ITWO .NE. 0) I5TWO = IBNDG2(5,ITWO)

      IF (I5TWO .EQ. I5ONE) THEN
         IBNDG2(5,NBNDG2) = I5ONE
      ELSE
         IF (I5TWO .EQ. 2) THEN
            IBNDG2(5,NBNDG2) = I5ONE
         ELSE
            IBNDG2(5,NBNDG2) = I5TWO
         ENDIF
      ENDIF

      KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0003)
      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0003)

C     ONLY SOUTHERN EDGE WILL BE CHECKED FOR SPECIAL INTERPOLATION

      K3BOUN = IAND (KX,KL0F00)
      IF (K3BOUN .NE. 0) THEN
        KAUXG2(LMSW) = IOR (KAUXG2(LMSW),K3BOUN)
        KAUXG2(LMSE) = IOR (KAUXG2(LMSE),K3BOUN)
        IBNODE(1)    = KSE
        IBNODE(2)    = KSW
        IBNODE(3)    = ICELG2(4,LVSE)
        IBNODE(4)    = ICELG2(2,LVSW)
        IBNODE(5)    = KS
        INTERF       = ISHFT (K3BOUN,-8)
        CALL G2BPIN (IBNODE,INTERF)
      ENDIF
```

```
      GO TO 290

C     DIVIDED CELL WAS AT SOUTHEASTERN CORNER

220   IBNDG2(3,IONE)   = LMSW
      IBNDG2(2,ICOR)   = LMSE
      IBNDG2(2,ITWO)   = LMNE

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KS
      IBNDG2(2,NBNDG2) = LMSW
      IBNDG2(3,NBNDG2) = LMSE
      IBNDG2(4,NBNDG2) = 3
      IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
      NBCPG2(2,1)      = NBNDG2

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KE
      IBNDG2(2,NBNDG2) = LMSE
      IBNDG2(3,NBNDG2) = LMNE
      IBNDG2(4,NBNDG2) = 5
      IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
      NBCPG2(2,2)      = NBNDG2

      KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0003)
      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0007)
      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL0006)

      GO TO 290

C     DIVIDED CELL WAS ALONG EASTERN EDGE

230   IBNDG2(3,IONE)   = LMSE
      IBNDG2(2,ITWO)   = LMNE

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KE
      IBNDG2(2,NBNDG2) = LMSE
      IBNDG2(3,NBNDG2) = LMNE
      IBNDG2(4,NBNDG2) = 5
      IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)

      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0006)
      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL0006)

      GO TO 290

C     DIVIDED CELL WAS AT NORTHEASTERN CORNER

240   IBNDG2(3,IONE)   = LMSE
      IBNDG2(2,ICOR)   = LMNE
      IBNDG2(2,ITWO)   = LMNW

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KE
      IBNDG2(2,NBNDG2) = LMSE
```

```
      IBNDG2(3,NBNDG2) = LMNE
      IBNDG2(4,NBNDG2) = 5
      IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
      NBCPG2(3,1)      = NBNDG2

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KN
      IBNDG2(2,NBNDG2) = LMNE
      IBNDG2(3,NBNDG2) = LMNW
      IBNDG2(4,NBNDG2) = 7
      IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
      NBCPG2(3,2)      = NBNDG2

      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0006)
      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL000E)
      KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL000C)

      GO TO 290

C     DIVIDED CELL WAS ALONG NORTHERN EDGE

250   IBNDG2(3,IONE)   = LMNE
      IBNDG2(2,ITWO)   = LMNW

      NBNDG2           = NBNDG2+1
      IBNDG2(1,NBNDG2) = KN
      IBNDG2(2,NBNDG2) = LMNE
      IBNDG2(3,NBNDG2) = LMNW
      IBNDG2(4,NBNDG2) = 7

      I5ONE = IBNDG2(5,IONE)
      I5TWO = I5ONE
      IF (ITWO .NE. 0) I5TWO = IBNDG2(5,ITWO)

      IF (I5TWO .EQ. I5ONE) THEN
         IBNDG2(5,NBNDG2) = I5ONE
      ELSE
         IF (I5ONE .EQ. 2) THEN
            IBNDG2(5,NBNDG2) = I5TWO
         ELSE
            IBNDG2(5,NBNDG2) = I5ONE
         ENDIF
      ENDIF

      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL000C)
      KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL000C)

C     ONLY NORTHERN EDGE WILL BE CHECKED FOR SPECIAL INTERPOLATION

      K3BOUN = IAND (KX,KLOF00)
      IF (K3BOUN .NE. 0) THEN
        KAUXG2(LMNE) = IOR (KAUXG2(LMNE),K3BOUN)
        KAUXG2(LMNW) = IOR (KAUXG2(LMNW),K3BOUN)
        IBNODE(1)    = KNE
        IBNODE(2)    = KNW
        IBNODE(3)    = ICELG2(6,LVNE)
        IBNODE(4)    = ICELG2(8,LVNW)
```

```
          IBNODE(5)    = KN
          INTERF       = ISHFT (K3BOUN,-8)
          CALL G2BPIN (IBNODE,INTERF)
        ENDIF

        GO TO 290

C       DIVIDED CELL WAS AT NORTHWESTERN CORNER

260     IBNDG2(3,IONE)   = LMNE
        IBNDG2(2,ICOR)   = LMNW
        IBNDG2(2,ITWO)   = LMSW

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KN
        IBNDG2(2,NBNDG2) = LMNE
        IBNDG2(3,NBNDG2) = LMNW
        IBNDG2(4,NBNDG2) = 7
        IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
        NBCPG2(4,1)      = NBNDG2

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KW
        IBNDG2(2,NBNDG2) = LMNW
        IBNDG2(3,NBNDG2) = LMSW
        IBNDG2(4,NBNDG2) = 9
        IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
        NBCPG2(4,2)      = NBNDG2

        KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL000C)
        KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL000D)
        KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0009)

        GO TO 290

C       DIVIDED CELL WAS AT WESTERN EDGE

270     IBNDG2(2,ITWO)   = LMSW
        IF (IONE .NE. 0) THEN
           IBNDG2(3,IONE)   = LMNW
        ELSE IF (ICOR .NE. 0) THEN
           IBNDG2(2,ICOR)   = LMNW
        ENDIF

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KW
        IBNDG2(2,NBNDG2) = LMNW
        IBNDG2(3,NBNDG2) = LMSW
        IBNDG2(4,NBNDG2) = 9
        IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)

        KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL0009)
        KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0009)

        GO TO 290

C       ERROR IN BOUNDARY CELL POINTERS
```

```
280     ZER1 = LCELL
        ZER2 = KX
        CALL ERRORM (11,'G2DIVO','LCELL ',ZER1,'KX    ',ZER2,JPRINT,
     1      'ERROR IN BOUNDARY NODE CALCULATION')
290     CONTINUE

C       COMPUTE THE METRICS ETC. FOR THE NEWLY CREATED CELLS

        CALL M2AREA (LMSW)
        CALL M2AREA (LMSE)
        CALL M2AREA (LMNE)
        CALL M2AREA (LMNW)
C
C       CHECK IF THE CELL HAS FUEL INJECTED TO IT
C
        IF (IAND(KX,KL1000) .EQ. 0) RETURN
C
C       ONLY THE CELLS WHICH ARE VERTICALLY ALLIGNED AND WHICH ARE ON
C       THE RIGHT HAND SIDE OF THE PLANE OF INJECTION ARE MARKED
C
        KAUXG2(LMSW)   = IOR(KAUXG2(LMSW),KL1000)
        KAUXG2(LMSE)   = IOR(KAUXG2(LMSE),KL1000)
        KAUXG2(LMNE)   = IOR(KAUXG2(LMNE),KL1000)
        KAUXG2(LMNW)   = IOR(KAUXG2(LMNW),KL1000)

        if (levelg .eq. 0) then
           nbndg2 = nbndg2 + 1
           ibndg2(1,nbndg2) = ks
           ibndg2(2,nbndg2) = 0
           ibndg2(3,nbndg2) = 0
           ibndg2(4,nbndg2) = 3
           ibndg2(5,nbndg2) = 11

           nbndg2 = nbndg2 + 1
           ibndg2(1,nbndg2) = ke
           ibndg2(2,nbndg2) = 0
           ibndg2(3,nbndg2) = 0
           ibndg2(4,nbndg2) = 5
           ibndg2(5,nbndg2) = 2

           nbndg2 = nbndg2 + 1
           ibndg2(1,nbndg2) = kn
           ibndg2(2,nbndg2) = 0
           ibndg2(3,nbndg2) = 0
           ibndg2(4,nbndg2) = 7
           ibndg2(5,nbndg2) = 11

           nbndg2 = nbndg2 + 1
           ibndg2(1,nbndg2) = kw
           ibndg2(2,nbndg2) = 0
           ibndg2(3,nbndg2) = 0
           ibndg2(4,nbndg2) = 9
           ibndg2(5,nbndg2) = 11

        else
```

```
ibfsw = 0
ibfse = 0
ibfne = 0
ibfnw = 0

do ibnd = 1, nbndg2
   if (ibndg2(1,ibnd) .eq. ksw) ibfsw = ibnd
   if (ibndg2(1,ibnd) .eq. kse) ibfse = ibnd
   if (ibndg2(1,ibnd) .eq. kne) ibfne = ibnd
   if (ibndg2(1,ibnd) .eq. knw) ibfnw = ibnd
enddo

  if (ibfsw .eq. 0) then
     nbndg2 = nbndg2 + 1
     ibndg2(1,nbndg2) = ke
     ibndg2(2,nbndg2) = 0
     ibndg2(3,nbndg2) = 0
     ibndg2(4,nbndg2) = 5
     ibndg2(5,nbndg2) = 2

     nbndg2 = nbndg2 + 1
     ibndg2(1,nbndg2) = kn
     ibndg2(2,nbndg2) = 0
     ibndg2(3,nbndg2) = 0
     ibndg2(4,nbndg2) = 7
     ibndg2(5,nbndg2) = 11
  else if (ibfse .eq. 0) then
     nbndg2 = nbndg2 + 1
     ibndg2(1,nbndg2) = kn
     ibndg2(2,nbndg2) = 0
     ibndg2(3,nbndg2) = 0
     ibndg2(4,nbndg2) = 7
     ibndg2(5,nbndg2) = 11

     nbndg2 = nbndg2 + 1
     ibndg2(1,nbndg2) = kw
     ibndg2(2,nbndg2) = 0
     ibndg2(3,nbndg2) = 0
     ibndg2(4,nbndg2) = 9
     ibndg2(5,nbndg2) = 11
  else if (ibfne .eq. 0) then
     nbndg2 = nbndg2 + 1
     ibndg2(1,nbndg2) = kw
     ibndg2(2,nbndg2) = 0
     ibndg2(3,nbndg2) = 0
     ibndg2(4,nbndg2) = 9
     ibndg2(5,nbndg2) = 11

     nbndg2 = nbndg2 + 1
     ibndg2(1,nbndg2) = ks
     ibndg2(2,nbndg2) = 0
     ibndg2(3,nbndg2) = 0
     ibndg2(4,nbndg2) = 3
     ibndg2(5,nbndg2) = 11
  else if (ibfnw .eq. 0) then
     nbndg2 = nbndg2 + 1
     ibndg2(1,nbndg2) = ks
```

```
                ibndg2(2,nbndg2) = 0
              -  ibndg2(3,nbndg2) = 0
               - ibndg2(4,nbndg2) = 3
                ibndg2(5,nbndg2) = 11

                nbndg2 = nbndg2 + 1
                ibndg2(1,nbndg2) = ke
                ibndg2(2,nbndg2) = 0
                ibndg2(3,nbndg2) = 0
                ibndg2(4,nbndg2) = 5
                ibndg2(5,nbndg2) = 2
             else
                write(6,*) ' heyman error g2divu'
             endif
          endif

c         NUMDH2          = NUMDH2 + 1
c         NODEH2(NUMDH2) = KW

          RETURN
          END
```

G2DIV0

```
          SUBROUTINE G2DIV0 (LCELL, IWARN)

          INCLUDE '[.INC] PRECIS.INC /LIST'
          INCLUDE '[.INC] PARMV2.INC /LIST'
          INCLUDE '[.INC] G2COMN.INC /LIST'
          INCLUDE '[.INC] HEXCOD.INC      '
          INCLUDE '[.INC] IOCOMN.INC /LIST'
          DIMENSION IBNODE(5)
          LOGICAL  IWRITE
C
C*****************************************************************
C
C         THIS SUBROUTINE DIVIDES CELL 'LCELL' INTO FOUR SMALLER CELLS
C         AND PERFORMS ALL NECESSARY POINTER SYSTEM REALIGNMENTS
C
C
C         SPECIAL EXPLANATION OF AUXILIARY CELL POINTERS
C
C           KAUXG2(LCELL) HAS THE HEXIDECIMAL FORM:
C
C                       'X  X  X  X  X  X  X  X '
C                         8  7  6  5  4  3  2  1
C
C            WHERE:
C                X    INDICATES THAT CELL IS A BOUNDARY CELL
C                 1
C
C                X    INDICATES THAT THE CELL WAS RECENTLY DIVIDED
```

```
C              2       AND HENCE MUST NOT BE COLLAPSED
C
C            X    INDICATES THE BOUNDARY INTERPOLATION FUNCTION TYPE
C             3
C
C            X    INDICATES SPECIAL CELLS (E.G., FUEL INJECTION CELLS)
C             4
C
C            X    INDICATES THE SPATIAL LEVEL OF THE  CELL ( <= MLVLG2)
C             5
C
C         EXPLANATION OF HISTORY POINTER OF THE CELLS (BYTE 2)
C
C         THE CELL WHICH IS DIVIDED CAN ONLY BE COLLAPSED AFTER THREE
C         GENERATIONS LATER, NOTE THAT INCHIS=48=2^5+2^4 (I.E., THE
C         SECOND BYTE IS SET EQUAL TO 3).  AFTER EACH ADAPTATION CYCLE
C         (I.E., A CALL TO A2MTHO) THIS POINTER IS REDUCED BY ONE UNTIL
C         IT BECOMES ZERO.
C
C         EXPLANATION OF SPATIAL LEVEL POINTER OF THE CELLS (BYTE 5)
C
C         K5LEVG : 5TH BYTE OF THE GIVEN CELL LCELL
C         K5LEVN : 5TH BYTE OF THE NEW CELLS
C         LEVELG : LEVEL OF THE GIVEN CELL LCELL
C         LEVELN : LEVEL OF THE NEW CELLS (TO BE DIVIDED, LEVELG+1)
C
C*********************************************************************
C
        INCHIS = 48
        MPOINT = 10
        KVCORR = 2**16
        NADCEL = 4
C
C       --------------
C       OVERFLOW CHECK
C       --------------
C
C       CHECK FOR OVERFLOW IN NODE ARRAYS
C
        IF(NNODG2+5 .GT. MNODG2) THEN
          ZER1 = MNODG2
          ZER2 = NNODG2
          CALL WARNIN (6,'G2DIVO','MNODG2',ZER1,'NNODG2',ZER2,JPRINT,
     1      'NUMBER OF NODES EXCEEDS ITS LIMIT')
          IWARN = 6
          RETURN
        ENDIF

C       CHECK FOR OVERFLOW IN CELL LIMIT

        IF (NCELG2+4 .GT. MCELG2) THEN
          ZER1 = MCELG2
          ZER2 = NCELG2
          CALL WARNIN (7,'G2DIVO','MCELG2',ZER1,'NCELG2',ZER2,JPRINT,
     1      'NUMBER OF CELLS EXCEEDS ITS LIMIT')
          IWARN = 7
          RETURN
```

```
      ENDIF

C     CHECK FOR OVERFLOW IN BOUNDARY CONDITION ARRAY

      IF (NBNDG2+2 .GT. MBNDG2) THEN
        ZER1 = MBNDG2
        ZER2 = NBNDG2
        CALL WARNIN (8,'G2DIVO','MBNDG2',ZER1,'NBNDG2',ZER2,JPRINT,
     1    'NUMBER OF BOUNDARY NODES EXCEEDS ITS LIMIT')
        IWARN = 8
        RETURN
      ENDIF

C     FIND THE LEVEL LEVELG OF THE GIVEN CELL

      KX     = KAUXG2(LCELL)
      K5LEVG = IAND(KX,KU000F)
      LEVELG = ISHFT(K5LEVG,-16)

C     HAS CELL BEEN PREVIOUSLY DIVIDED
C     THE FOLLOWING ERROR CONDITION IS NOT REALLY NEEDED

      KC     = ICELG2(1,LCELL)

      IF (KC .NE. 0) THEN
        ZER1   = LCELL
        ZER2   = KC
        CALL WARNIN (9,'G2DIVO','LCELL ',ZER1,' KC   ',ZER2,JPRINT,
     1    'THE CELL IS ALREADY DIVIDED ')
        IWARN = 9
        RETURN
      ENDIF
C
C     --------------
C     POINTER SAVING
C     --------------
C
C     SAVE REST OF CELL POINTERS

      KSW    = ICELG2(2,LCELL)
      KS     = ICELG2(3,LCELL)
      KSE    = ICELG2(4,LCELL)
      KE     = ICELG2(5,LCELL)
      KNE    = ICELG2(6,LCELL)
      KN     = ICELG2(7,LCELL)
      KNW    = ICELG2(8,LCELL)
      KW     = ICELG2(9,LCELL)
      K5LEVN = K5LEVG + KVCORR
      LEVELN = LEVELG + 1
      MAXLEV = MAX(NLVLG2, LEVELN)

      IF (LEVELN .GT. MALVG2) THEN
        ZER1 = LEVELN
        ZER2 = MALVG2
C       CALL WARNIN (10,'G2DIVO','LEVELN ',ZER1,'MALVG2',ZER2,JPRINT,
C    1    'NEW LEVEL INCREASES BEYOND THE GIVEN LIMIT')
        IWARN = 10
```

```
        RETURN
       ENDIF
C
C      ----------------------
C      NEIGHBOUR DETERMINATION
C      ----------------------
C
C      FIND CELLS WHICH BOUND DIVIDED CELL
C
C       |------------|---------------|------------|
C       |            |               |            |  K FOR NODE
C       |      + + + + + + + + + + + + + + +      |  L FOR CELL
C       |      +LCNW | LHNW  + LHNE  |LCNE +      |
C       |      +     |       +       |     +      |
C       |------+-----+-------+-------+-----+------|
C       |      +     |KNW   KN    KNE|     +      | M:CENTER (MIDDLE)
C       |      +LVNW | LMNW     LMNE |LVNE +      | C:CORNER (ADJACENT)
C       |      + + + +KW   LCELL   KE+ + + +      | H:HORIZONTAL (ADJAC)
C       |      +LVSW | LMSW     LMSE |LVSE +      | V:VERTICAL (ADJACENT)
C       |      +     |KSW   KS    KSE|     +      |
C       |------+-----+-------+-------+-----+------|
C       |      +     |       +       |     +      |
C       |      +LCSW | LHSW  + LHSE  |LCSE +      |
C       |      + + + + + + + + + + + + + + +      |
C       |            |               |            |
C       |------------|---------------|------------|
C
       LVSW   = NEIBG2(4,KSW)
       LCSW   = NEIBG2(1,KSW)
       LHSW   = NEIBG2(2,KSW)
       LHSE   = NEIBG2(1,KSE)
       LCSE   = NEIBG2(2,KSE)
       LVSE   = NEIBG2(3,KSE)
       LVNE   = NEIBG2(2,KNE)
       LCNE   = NEIBG2(3,KNE)
       LHNE   = NEIBG2(4,KNE)
       LHNW   = NEIBG2(3,KNW)
       LCNW   = NEIBG2(4,KNW)
       LVNW   = NEIBG2(1,KNW)
C
C      ----------------------
C      LEVEL DIFFERENTIAL CHECK
C      ----------------------
C
C      IF THE COMPONENT CELLS ARE JUST INSIDE EMBEDDED REGION THEN
C      THEY CAN NOT BE DIVIDED; THIS WILL BE SO IF THE LEVELS OF
C      THE NEIGHBOURHOOD CELLS DIFFER BY MORE THAN ONE
C      FIRST DO THE CORNER CELLS

       IF (LCSW .NE. 0) THEN
         K5LCOR = IAND(KAUXG2(LCSW),KU000F)
         LEVELC = ISHFT(K5LCOR,-16)
         IDFL   = LEVELC-LEVELG
         IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
       ENDIF
C
       IF (LCSE .NE. 0) THEN
```

```
          K5LCOR = IAND(KAUXG2(LCSE),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
        ENDIF
C
        IF (LCNE .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCNE),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
        ENDIF
C
        IF (LCNW .NE. 0) THEN
          K5LCOR = IAND(KAUXG2(LCNW),KU000F)
          LEVELC = ISHFT(K5LCOR,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
        ENDIF
C
C       NOW DO EDGE CELLS
C
        IF (LHSW .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LHSW),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
        ENDIF
C
        IF (LVSE .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LVSE),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
        ENDIF
C
        IF (LHNE .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LHNE),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
        ENDIF
C
        IF (LVNW .NE. 0) THEN
          K5LEDG = IAND(KAUXG2(LVNW),KU000F)
          LEVELC = ISHFT(K5LEDG,-16)
          IDFL   = LEVELC-LEVELG
          IF (IDFL .LT. 0 .OR. IDFL .GT. 1) RETURN
        ENDIF
C
C       ----------------------
C       NODE ASSIGNMENTS CHECK
C       ----------------------
C
C       CHECK IF THE CELLS AGREE ON SOME OF THE NODE ASSIGNMENTS
C       THIS WILL BE REMOVED LATTER; ONCE THE CODE IS TESTED
C       FIRST CHECK CORNER CELLS
```

```
      IF (LCSW .NE. 0) THEN
       IF (ICELG2(6,LCSW) .NE. KSW) THEN
         ZER1 = ICELG2(6,LCSW)
         ZER2 = KSW
         CALL ERRORM (12,'G2DIVO','KSWCAL',ZER1,'KSW   ',ZER2,JPRINT,
     1     'ERROR IN NODE ASSIGNMENT')
       ENDIF
      ENDIF
C
      IF (LCSE .NE. 0) THEN
       IF (ICELG2(8,LCSE) .NE. KSE) THEN
         ZER1 = ICELG2(8,LCSE)
         ZER2 = KSE
         CALL ERRORM (12,'G2DIVO','KSECAL',ZER1,'KSE   ',ZER2,JPRINT,
     1     'ERROR IN NODE ASSIGNMENT')
       ENDIF
      ENDIF
C
      IF (LCNE .NE. 0) THEN
       IF (ICELG2(2,LCNE) .NE. KNE) THEN
         ZER1 = ICELG2(2,LCNE)
         ZER2 = KNE
         CALL ERRORM (12,'G2DIVO','KNECAL',ZER1,'KNE   ',ZER2,JPRINT,
     1     'ERROR IN NODE ASSIGNMENT')
       ENDIF
      ENDIF
C
      IF (LCNW .NE. 0) THEN
       IF (ICELG2(4,LCNW) .NE. KNW) THEN
         ZER1 = ICELG2(4,LCNW)
         ZER2 = KNW
         CALL ERRORM (12,'G2DIVO','KNWCAL',ZER1,'KNW   ',ZER2,JPRINT,
     1     'ERROR IN NODE ASSIGNMENT')
       ENDIF
      ENDIF
C
C     NOW CHECK HORIZONTAL AND VERTICAL ADJACENT CELLS
C
      IF (KS .NE. 0) THEN
       IF (ICELG2(6,LHSW) .NE. KS) THEN
         ZER1 = ICELG2(6,LHSW)
         ZER2 = KS
         CALL ERRORM (12,'G2DIVO','KS-CAL',ZER1,'KS    ',ZER2,JPRINT,
     1     'ERROR IN NODE ASSIGNMENT')
       ENDIF
       IF (ICELG2(8,LHSE) .NE. KS) THEN
         ZER1 = ICELG2(8,LHSE)
         ZER2 = KS
         CALL ERRORM (12,'G2DIVO','KS-CAL',ZER1,'KS    ',ZER2,JPRINT,
     1     'ERROR IN NODE ASSIGNMENT')
       ENDIF
      ENDIF
C
      IF (KE .NE. 0) THEN
       IF (ICELG2(8,LVSE) .NE. KE) THEN
         ZER1 = ICELG2(8,LVSE)
```

```
            ZER2 = KE
            CALL ERRORM (12,'G2DIVO','KE-CAL',ZER1,'KE    ',ZER2,JPRINT,
     1        'ERROR IN NODE ASSIGNMENT')
           ENDIF
           IF (ICELG2(2,LVNE) .NE. KE) THEN
            ZER1 = ICELG2(2,LVNE)
            ZER2 = KE
            CALL ERRORM (12,'G2DIVO','KE-CAL',ZER1,'KE    ',ZER2,JPRINT,
     1        'ERROR IN NODE ASSIGNMENT')
           ENDIF
          ENDIF
C
          IF (KN .NE. 0) THEN
           IF (ICELG2(2,LHNE) .NE. KN) THEN
            ZER1 = ICELG2(2,LHNE)
            ZER2 = KN
            CALL ERRORM (12,'G2DIVO','KN-CAL',ZER1,'KN    ',ZER2,JPRINT,
     1        'ERROR IN NODE ASSIGNMENT')
           ENDIF
           IF (ICELG2(4,LHNW) .NE. KN) THEN
            ZER1 = ICELG2(4,LHNW)
            ZER2 = KN
            CALL ERRORM (12,'G2DIVO','KN-CAL',ZER1,'KN    ',ZER2,JPRINT,
     1        'ERROR IN NODE ASSIGNMENT')
           ENDIF
          ENDIF
C
          IF (KW .NE. 0) THEN
           IF (ICELG2(4,LVNW) .NE. KW) THEN
            ZER1 = ICELG2(4,LVNW)
            ZER2 = KW
            CALL ERRORM (12,'G2DIVO','KW-CAL',ZER1,'KW    ',ZER2,JPRINT,
     1        'ERROR IN NODE ASSIGNMENT')
           ENDIF
           IF (ICELG2(6,LVSW) .NE. KW) THEN
            ZER1 = ICELG2(6,LVSW)
            ZER2 = KW
            CALL ERRORM (12,'G2DIVO','KW-CAL',ZER1,'KW    ',ZER2,JPRINT,
     1        'ERROR IN NODE ASSIGNMENT')
           ENDIF
          ENDIF
C
C         -----------
C         DEBUG PRINT
C         -----------
C
C         PRINT OUT PARAMETERS BEFORE DIVISION
C
          IWRITE  = IDBGG2 .EQ. 2 .OR. IDBGG2 .GT. 1000
C
          IF (IWRITE)  THEN
             WRITE(JDEBUG,1000)
             WRITE(JDEBUG,1100)
             WRITE(JDEBUG,1200)
             WRITE(JDEBUG,1300)
C
C            GENERAL INFORMATION
```

```
C
      WRITE(JDEBUG,1400) NNODG2, NCELG2, NBNDG2, LEVELG
C
C     POINTERS OF MAIN CELL LCELL
C
      WRITE(JDEBUG,1500) LCELL, KC , KSW, KS , KSE, KE,
     1                           KNE, KN , KNW, KW , KX
      WRITE(JDEBUG,1600) (ICELG2(I,LCELL), I = 1, 10)
C
C     NEIGHBOUR CELLS AND THEIR POINTERS
C
      WRITE(JDEBUG,1700) LVSW, LCSW, LHSW, LHSE, LCSE, LVSE,
     1                   LVNE, LCNE, LHNE, LHNW, LCNW, LVNW
C
      IF (LVSW .NE. 0) THEN
         WRITE(JDEBUG,1800) (ICELG2(I,LVSW), I = 1, 10)
      ENDIF
      IF (LCSW .NE. 0) THEN
         WRITE(JDEBUG,1900) (ICELG2(I,LCSW), I = 1, 10)
      ENDIF
      IF (LHSW .NE. 0) THEN
         WRITE(JDEBUG,2000) (ICELG2(I,LHSW), I = 1, 10)
      ENDIF
C
      IF (LHSE .NE. 0) THEN
         WRITE(JDEBUG,2100) (ICELG2(I,LHSE), I = 1, 10)
      ENDIF
      IF (LCSE .NE. 0) THEN
         WRITE(JDEBUG,2200) (ICELG2(I,LCSE), I = 1, 10)
      ENDIF
      IF (LVSE .NE. 0) THEN
         WRITE(JDEBUG,2300) (ICELG2(I,LVSE), I = 1, 10)
      ENDIF
C
      IF (LVNE .NE. 0) THEN
         WRITE(JDEBUG,2400) (ICELG2(I,LVNE), I = 1, 10)
      ENDIF
      IF (LCNE .NE. 0) THEN
         WRITE(JDEBUG,2500) (ICELG2(I,LCNE), I = 1, 10)
      ENDIF
      IF (LHNE .NE. 0) THEN
         WRITE(JDEBUG,2600) (ICELG2(I,LHNE), I = 1, 10)
      ENDIF
C
      IF (LHNW .NE. 0) THEN
         WRITE(JDEBUG,2700) (ICELG2(I,LHNW), I = 1, 10)
      ENDIF
      IF (LCNW .NE. 0) THEN
         WRITE(JDEBUG,2800) (ICELG2(I,LCNW), I = 1, 10)
      ENDIF
      IF (LVNW .NE. 0) THEN
         WRITE(JDEBUG,2900) (ICELG2(I,LVNW), I = 1, 10)
      ENDIF
C
C     NEIGHBOURING CELLS OF ALL NODES OF LCELL
C
      WRITE(JDEBUG,3000) (NEIBG2(I,KSW),I=1,4)
```

```
            WRITE(JDEBUG,3100) (NEIBG2(I,KSE),I=1,4)
            WRITE(JDEBUG,3200) (NEIBG2(I,KNE),I=1,4)
            WRITE(JDEBUG,3300) (NEIBG2(I,KNW),I=1,4)
            IF (KS .NE. 0) THEN
               WRITE(JDEBUG,3400) (NEIBG2(I,KS),I=1,4)
            ENDIF
            IF (KE .NE. 0) THEN
               WRITE(JDEBUG,3500) (NEIBG2(I,KE),I=1,4)
            ENDIF
            IF (KN .NE. 0) THEN
               WRITE(JDEBUG,3600) (NEIBG2(I,KN),I=1,4)
            ENDIF
            IF (KW .NE. 0) THEN
               WRITE(JDEBUG,3700) (NEIBG2(I,KW),I=1,4)
            ENDIF
C
         ENDIF       ! IWRITE
C
C        ----------------
C        DIVISION PROCESS
C        ----------------
C
C        ALL ERRORS (EXCEPT BOUNDARY NODES) HAVE BEEN CHECKED, SO
C        INITIATE THE PROCESS OF CELL DIVISION
C        UPDATE THE OVERALL MAXIMUM LEVEL POINTER
C
         NLVLG2 = MAXLEV

C        UPDATE THE NUMBER OF CELLS AT THE NEW LEVEL

         ILVLG2(3,LEVELN) = ILVLG2(3,LEVELN) + NADCEL

C        CREATE NODE AT CENTER OF CELL

         NNODG2 = NNODG2 + 1
         KC     = NNODG2

C        COMPUTE THE GEOMETRIC QUANTITIES AT THE NEW CENTER NODE

         GEOMG2(1,KC) = 0.25*( GEOMG2(1,KSW) + GEOMG2(1,KSE) +
     1                         GEOMG2(1,KNE) + GEOMG2(1,KNW) )
         GEOMG2(2,KC) = 0.25*( GEOMG2(2,KSW) + GEOMG2(2,KSE) +
     1                         GEOMG2(2,KNE) + GEOMG2(2,KNW) )

C        LINEAR INTERPOLATION FOR DEPENDENT VARIABLES

         DO 10 J = 1, NEQNFL
           DPENG2(J,KC) = 0.25*( DPENG2(J,KSW) + DPENG2(J,KSE) +
     1                           DPENG2(J,KNE) + DPENG2(J,KNW) )
10       CONTINUE
         PRESG2(KC) = 0.25*( PRESG2(KSW) + PRESG2(KSE) +
     1                       PRESG2(KNE) + PRESG2(KNW) )
         TEMPG2(KC) = 0.25*( TEMPG2(KSW) + TEMPG2(KSE) +
     1                       TEMPG2(KNE) + TEMPG2(KNW) )

C        DOES SOUTHERN NODE ALREADY EXIST; IF NOT CREATE IT
```

```
      IF (KS .EQ. 0) THEN
        NNODG2       = NNODG2 + 1
        KS           = NNODG2
        GEOMG2(1,KS) = 0.50*( GEOMG2(1,KSW) + GEOMG2(1,KSE) )
        GEOMG2(2,KS) = 0.50*( GEOMG2(2,KSW) + GEOMG2(2,KSE) )
        IF (LHSW .NE. 0 .AND. LHSE .NE. 0) THEN
          NEIBG2(1,KS) = LHSW
          NEIBG2(2,KS) = LHSE
        ELSE
          NEIBG2(1,KS) = 0
          NEIBG2(2,KS) = 0
        ENDIF
        DO 20 J = 1, NEQNFL
          DPENG2(J,KS) = 0.50*( DPENG2(J,KSW) + DPENG2(J,KSE) )
20      CONTINUE
        PRESG2(KS) = 0.50*( PRESG2(KSW) + PRESG2(KSE) )
        TEMPG2(KS) = 0.50*( TEMPG2(KSW) + TEMPG2(KSE) )
      ENDIF

C     DOES EASTERN  NODE ALREADY EXIST; IF NOT CREATE IT

      IF (KE .EQ. 0) THEN
        NNODG2       = NNODG2 + 1
        KE           = NNODG2
        GEOMG2(1,KE) = 0.50*( GEOMG2(1,KNE) + GEOMG2(1,KSE) )
        GEOMG2(2,KE) = 0.50*( GEOMG2(2,KNE) + GEOMG2(2,KSE) )
        IF (LVSE .NE. 0 .AND. LVNE .NE. 0) THEN
          NEIBG2(2,KE) = LVSE
          NEIBG2(3,KE) = LVNE
        ELSE
          NEIBG2(2,KE) = 0
          NEIBG2(3,KE) = 0
        ENDIF
        DO 30 J = 1, NEQNFL
          DPENG2(J,KE) = 0.50*( DPENG2(J,KNE) + DPENG2(J,KSE) )
30      CONTINUE
        PRESG2(KE) = 0.50*( PRESG2(KNE) + PRESG2(KSE) )
        TEMPG2(KE) = 0.50*( TEMPG2(KNE) + TEMPG2(KSE) )
      ENDIF

C     DOES NORTHERN NODE ALREADY EXIST; IF NOT CREATE IT

      IF (KN .EQ. 0) THEN
        NNODG2       = NNODG2 + 1
        KN           = NNODG2
        GEOMG2(1,KN) = 0.50*( GEOMG2(1,KNE) + GEOMG2(1,KNW) )
        GEOMG2(2,KN) = 0.50*( GEOMG2(2,KNE) + GEOMG2(2,KNW) )
        IF (LHNW .NE. 0 .AND. LHNE .NE. 0) THEN
          NEIBG2(3,KN) = LHNE
          NEIBG2(4,KN) = LHNW
        ELSE
          NEIBG2(3,KN) = 0
          NEIBG2(4,KN) = 0
        ENDIF
        DO 40 J = 1, NEQNFL
          DPENG2(J,KN) = 0.50*( DPENG2(J,KNE) + DPENG2(J,KNW) )
40      CONTINUE
```

```
          PRESG2(KN) = 0.50*( PRESG2(KNE) + PRESG2(KNW) )
          TEMPG2(KN) = 0.50*( TEMPG2(KNE) + TEMPG2(KNW) )
        ENDIF

C       DOES WESTERN NODE ALREADY EXIST; IF NOT CREATE IT

        IF (KW .EQ. 0) THEN
          NNODG2       = NNODG2 + 1
          KW           = NNODG2
          GEOMG2(1,KW) = 0.50*( GEOMG2(1,KNW) + GEOMG2(1,KSW) )
          GEOMG2(2,KW) = 0.50*( GEOMG2(2,KNW) + GEOMG2(2,KSW) )
          IF (LVSW .NE. 0 .AND. LVNW .NE. 0) THEN
            NEIBG2(1,KW) = LVSW
            NEIBG2(4,KW) = LVNW
          ELSE
            NEIBG2(1,KW) = 0
            NEIBG2(4,KW) = 0
          ENDIF
          DO 50 J = 1, NEQNFL
            DPENG2(J,KW) = 0.50*( DPENG2(J,KNW) + DPENG2(J,KSW) )
50        CONTINUE
          PRESG2(KW) = 0.50*( PRESG2(KNW) + PRESG2(KSW) )
          TEMPG2(KW) = 0.50*( TEMPG2(KNW) + TEMPG2(KSW) )
        ENDIF

C       UPDATE THE DIVIDED CELL -- NEW NODE

        ICELG2(1,LCELL) = KC
        ICELG2(3,LCELL) = KS
        ICELG2(5,LCELL) = KE
        ICELG2(7,LCELL) = KN
        ICELG2(9,LCELL) = KW

C       CREATE THE NEW CELLS

        LMSW   = NCELG2 + 1
        LMSE   = LMSW   + 1
        LMNE   = LMSE   + 1
        LMNW   = LMNE   + 1
        NCELG2 = NCELG2 + NADCEL

        ICELG2( 1,LMSW) = 0
        ICELG2( 2,LMSW) = KSW
        ICELG2( 3,LMSW) = 0
        ICELG2( 4,LMSW) = KS
        ICELG2( 5,LMSW) = 0
        ICELG2( 6,LMSW) = KC
        ICELG2( 7,LMSW) = 0
        ICELG2( 8,LMSW) = KW
        ICELG2( 9,LMSW) = 0
        ICELG2(10,LMSW) = LCELL
        KAUXG2(   LMSW) = K5LEVN + INCHIS

        ICELG2( 1,LMSE) = 0
        ICELG2( 2,LMSE) = KS
        ICELG2( 3,LMSE) = 0
        ICELG2( 4,LMSE) = KSE
```

```
      ICELG2( 5,LMSE) = 0
      ICELG2( 6,LMSE) = KE
      ICELG2( 7,LMSE) = 0
      ICELG2( 8,LMSE) = KC
      ICELG2( 9,LMSE) = 0
      ICELG2(10,LMSE) = LCELL
      KAUXG2(   LMSE) = K5LEVN + INCHIS

      ICELG2( 1,LMNE) = 0
      ICELG2( 2,LMNE) = KC
      ICELG2( 3,LMNE) = 0
      ICELG2( 4,LMNE) = KE
      ICELG2( 5,LMNE) = 0
      ICELG2( 6,LMNE) = KNE
      ICELG2( 7,LMNE) = 0
      ICELG2( 8,LMNE) = KN
      ICELG2( 9,LMNE) = 0
      ICELG2(10,LMNE) = LCELL
      KAUXG2(   LMNE) = K5LEVN + INCHIS

      ICELG2( 1,LMNW) = 0
      ICELG2( 2,LMNW) = KW
      ICELG2( 3,LMNW) = 0
      ICELG2( 4,LMNW) = KC
      ICELG2( 5,LMNW) = 0
      ICELG2( 6,LMNW) = KN
      ICELG2( 7,LMNW) = 0
      ICELG2( 8,LMNW) = KNW
      ICELG2( 9,LMNW) = 0
      ICELG2(10,LMNW) = LCELL
      KAUXG2(   LMNW) = K5LEVN + INCHIS

C     SET EDGE NODE POINTERS OF ALL NEIGHBOURING CELLS

      IF (LHSW.NE.0 .AND. LHSW.EQ.LHSE) ICELG2( 7,LHSW) = KS
      IF (LVSE.NE.0 .AND. LVSE.EQ.LVNE) ICELG2( 9,LVSE) = KE
      IF (LHNE.NE.0 .AND. LHNE.EQ.LHNW) ICELG2( 3,LHNE) = KN
      IF (LVNW.NE.0 .AND. LVNW.EQ.LVSW) ICELG2( 5,LVNW) = KW

C     UPDATE NEIGHBOUR-NODE-ARRAY

      NEIBG2(1,KC ) = LMSW
      NEIBG2(2,KC ) = LMSE
      NEIBG2(3,KC ) = LMNE
      NEIBG2(4,KC ) = LMNW
      NEIBG2(3,KSW) = LMSW
      NEIBG2(3,KS ) = LMSE
      NEIBG2(4,KS ) = LMSW
      NEIBG2(4,KSE) = LMSE
      NEIBG2(1,KE ) = LMSE
      NEIBG2(4,KE ) = LMNE
      NEIBG2(1,KNE) = LMNE
      NEIBG2(1,KN ) = LMNW
      NEIBG2(2,KN ) = LMNE
      NEIBG2(2,KNW) = LMNW
      NEIBG2(2,KW ) = LMSW
      NEIBG2(3,KW ) = LMNW
```

```
C
C       ---------------------
C       BOUNDARY NODE POINTERS
C       ---------------------
C
C       SKIP NEXT SECTION IF LCELL IS NOT A BOUNDARY CELL
C
        IGOTO = IAND(KX,KL000F)
        IF(IGOTO .EQ. 0) GO TO 290
C
C       SCAN THROUGH ALL BOUNDARY CONDITION POINTERS, LOOKING FOR POINTERS
C       TO THE DIVIDED CELL
C
        IONE = 0
        ICOR = 0
        ITWO = 0
        DO 60 IBND = 1, NBNDG2
           IF (IBNDG2(3,IBND) .EQ. LCELL      ) IONE = IBND
           IF (IBNDG2(2,IBND) .EQ. LCELL .AND.
     1         IBNDG2(3,IBND) .EQ. 0          ) ICOR = IBND
           IF (IBNDG2(2,IBND) .EQ. LCELL .AND.
     1         IBNDG2(3,IBND) .NE. 0          ) ITWO = IBND
60      CONTINUE
C
C       ERROR IF LEFT AND RIGHT POINTERS ARE NOT FOUND
C
        IF (IONE .EQ. 0 .OR. ITWO .EQ. 0) GO TO 280
C
C       PRINT OUT PARAMETERS FOR BOUNDARY NODES
C
        IF (IWRITE)  THEN
           WRITE(JDEBUG,3800) IONE, ICOR, ITWO
           WRITE(JDEBUG,3900) (IBNDG2(I,IONE),I=1,5)
           WRITE(JDEBUG,4000) (IBNDG2(I,ITWO),I=1,5)
           IF (ICOR .NE. 0) THEN
              WRITE(JDEBUG,4100) (IBNDG2(I,ICOR),I=1,5)
           ENDIF
        ENDIF       ! IWRITE
C
C
C               260        250        240
C                   +-------+-------+
C                   |13D   12C   14E|
C               270 +9   KAUXG2    6+ 230
C                   |11B    3      7|         280 : ERROR
C                   +-------+-------+
C               200        210        220     GO TO STATEMENTS
C
C
C       BRANCH OUT DEPENDING ON BOUNDARY TYPE
C
        GO TO (280,280,280,210,280,280,230,220,
     1         280,270,280,200,250,260,240,280), (IGOTO + 1)
        GO TO 280

C       DIVIDED CELL WAS AT SOUTHWESTERN CORNER
```

```
200     IBNDG2(2,ICOR)   = LMSW
        IBNDG2(2,ITWO)   = LMSE
        IBNDG2(3,IONE)   = LMNW

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KS
        IBNDG2(2,NBNDG2) = LMSW
        IBNDG2(3,NBNDG2) = LMSE
        IBNDG2(4,NBNDG2) = 3
        IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
        NBCPG2(1,2)      = NBNDG2

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KW
        IBNDG2(2,NBNDG2) = LMNW
        IBNDG2(3,NBNDG2) = LMSW
        IBNDG2(4,NBNDG2) = 9
        IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
        NBCPG2(1,1)      = NBNDG2

        KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL0009)
        KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL000B)
        KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0003)

        GO TO 290

C       DIVIDED CELL WAS ALONG SOUTHERN EDGE

210     IBNDG2(3,IONE)   = LMSW
        IBNDG2(2,ITWO)   = LMSE

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KS
        IBNDG2(2,NBNDG2) = LMSW
        IBNDG2(3,NBNDG2) = LMSE
        IBNDG2(4,NBNDG2) = 3
        IF (ITWO .NE. 0) THEN
           IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
        ELSE
           IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
        ENDIF

        KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0003)
        KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0003)
C       ONLY SOUTHERN EDGE WILL BE CHECKED FOR SPECIAL INTERPOLATION

        K3BOUN = IAND (KX,KLOFOO)
        IF (K3BOUN .NE. 0) THEN
          KAUXG2(LMSW) = IOR (KAUXG2(LMSW),K3BOUN)
          KAUXG2(LMSE) = IOR (KAUXG2(LMSE),K3BOUN)
          IBNODE(1)    = KSE
          IBNODE(2)    = KSW
          IBNODE(3)    = ICELG2(4,LVSE)
          IBNODE(4)    = ICELG2(2,LVSW)
          IBNODE(5)    = KS
          INTERF       = ISHFT (K3BOUN,-8)
          CALL G2BPIN (IBNODE,INTERF)
```

```
      ENDIF

      GO TO 290

C     DIVIDED CELL WAS AT SOUTHEASTERN CORNER

220   IBNDG2(3,IONE)   = LMSW
      IBNDG2(2,ICOR)   = LMSE
      IBNDG2(2,ITWO)   = LMNE

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KS
      IBNDG2(2,NBNDG2) = LMSW
      IBNDG2(3,NBNDG2) = LMSE
      IBNDG2(4,NBNDG2) = 3
      IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
      NBCPG2(2,1)      = NBNDG2

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KE
      IBNDG2(2,NBNDG2) = LMSE
      IBNDG2(3,NBNDG2) = LMNE
      IBNDG2(4,NBNDG2) = 5
      IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
      NBCPG2(2,2)      = NBNDG2

      KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0003)
      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0007)
      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL0006)

      GO TO 290

C     DIVIDED CELL WAS ALONG EASTERN EDGE

230   IBNDG2(3,IONE)   = LMSE
      IBNDG2(2,ITWO)   = LMNE

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KE
      IBNDG2(2,NBNDG2) = LMSE
      IBNDG2(3,NBNDG2) = LMNE
      IBNDG2(4,NBNDG2) = 5
      IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)

      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0006)
      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL0006)

      GO TO 290

C     DIVIDED CELL WAS AT NORTHEASTERN CORNER

240   IBNDG2(3,IONE)   = LMSE
      IBNDG2(2,ICOR)   = LMNE
      IBNDG2(2,ITWO)   = LMNW

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KE
```

```
      IBNDG2(2,NBNDG2) = LMSE
      IBNDG2(3,NBNDG2) = LMNE
      IBNDG2(4,NBNDG2) = 5
      IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
      NBCPG2(3,1)      = NBNDG2

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KN
      IBNDG2(2,NBNDG2) = LMNE
      IBNDG2(3,NBNDG2) = LMNW
      IBNDG2(4,NBNDG2) = 7
      IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
      NBCPG2(3,2)      = NBNDG2

      KAUXG2(LMSE)     = IOR(KAUXG2(LMSE),KL0006)
      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL000E)
      KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL000C)

      GO TO 290

C     DIVIDED CELL WAS ALONG NORTHERN EDGE

250   IBNDG2(3,IONE)   = LMNE
      IBNDG2(2,ITWO)   = LMNW

      NBNDG2           = NBNDG2+1
      IBNDG2(1,NBNDG2) = KN
      IBNDG2(2,NBNDG2) = LMNE
      IBNDG2(3,NBNDG2) = LMNW
      IBNDG2(4,NBNDG2) = 7
      IF (ITWO .NE. 0) THEN
         IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
      ELSE
         IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
      ENDIF

      KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL000C)
      KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL000C)

C     ONLY NORTHERN EDGE WILL BE CHECKED FOR SPECIAL INTERPOLATION

      K3BOUN = IAND (KX,KLOF00)
      IF (K3BOUN .NE. 0) THEN
        KAUXG2(LMNE) = IOR (KAUXG2(LMNE),K3BOUN)
        KAUXG2(LMNW) = IOR (KAUXG2(LMNW),K3BOUN)
        IBNODE(1)    = KNE
        IBNODE(2)    = KNW
        IBNODE(3)    = ICELG2(6,LVNE)
        IBNODE(4)    = ICELG2(8,LVNW)
        IBNODE(5)    = KN
        INTERF       = ISHFT (K3BOUN,-8)
        CALL G2BPIN (IBNODE,INTERF)
      ENDIF

      GO TO 290

C     DIVIDED CELL WAS AT NORTHWESTERN CORNER
```

```
260     IBNDG2(3,IONE)   = LMNE
        IBNDG2(2,ICOR)   = LMNW
        IBNDG2(2,ITWO)   = LMSW

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KN
        IBNDG2(2,NBNDG2) = LMNE
        IBNDG2(3,NBNDG2) = LMNW
        IBNDG2(4,NBNDG2) = 7
        IBNDG2(5,NBNDG2) = IBNDG2(5,IONE)
        NBCPG2(4,1)      = NBNDG2

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KW
        IBNDG2(2,NBNDG2) = LMNW
        IBNDG2(3,NBNDG2) = LMSW
        IBNDG2(4,NBNDG2) = 9
        IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)
        NBCPG2(4,2)      = NBNDG2

        KAUXG2(LMNE)     = IOR(KAUXG2(LMNE),KL000C)
        KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL000D)
        KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0009)

        GO TO 290

C       DIVIDED CELL WAS AT WESTERN EDGE

270     IBNDG2(3,IONE)   = LMNW
        IBNDG2(2,ITWO)   = LMSW

        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = KW
        IBNDG2(2,NBNDG2) = LMNW
        IBNDG2(3,NBNDG2) = LMSW
        IBNDG2(4,NBNDG2) = 9
        IBNDG2(5,NBNDG2) = IBNDG2(5,ITWO)

        KAUXG2(LMNW)     = IOR(KAUXG2(LMNW),KL0009)
        KAUXG2(LMSW)     = IOR(KAUXG2(LMSW),KL0009)

        GO TO 290

C       ERROR IN BOUNDARY CELL POINTERS

280     ZER1 = LCELL
        ZER2 = KX
        CALL ERRORM (11,'G2DIVO','LCELL ',ZER1,'KX    ',ZER2,JPRINT,
     1      'ERROR IN BOUNDARY NODE CALCULATION')
290     CONTINUE

C       COMPUTE THE METRICS ETC. FOR THE NEWLY CREATED CELLS

        CALL M2AREA (LMSW)
        CALL M2AREA (LMSE)
        CALL M2AREA (LMNE)
```

```
      CALL M2AREA (LMNW)
C
C          -
C
C     CHECK IF THE CELL HAS FUEL INJECTED TO IT
C
      IF (IAND(KX,KL1000) .EQ. 0) RETURN
C
C     ONLY THE CELLS WHICH ARE VERTICALLY ALLIGNED AND WHICH ARE ON
C     THE RIGHT HAND SIDE OF THE PLANE OF INJECTION ARE MARKED
C
      KAUXG2(LMSW)   = IOR(KAUXG2(LMSW),KL1000)
      KAUXG2(LMNW)   = IOR(KAUXG2(LMNW),KL1000)
      NUMDH2         = NUMDH2 + 1
      NODEH2(NUMDH2) = KW

C     -----------
C     DEBUG PRINT
C     -----------
C
C     PRINT OUT PARAMETERS AFTER DIVISION
C
      IF (IWRITE)  THEN
         WRITE(JDEBUG,4200)
C
C        GENERAL INFORMATION
C
         WRITE(JDEBUG,1400) NNODG2, NCELG2, NBNDG2, LEVELN
C
C        POINTERS OF MAIN CELL LCELL
C
         WRITE(JDEBUG,1500) LCELL, KC , KSW, KS , KSE, KE,
     1                              KNE, KN , KNW, KW , KX
         WRITE(JDEBUG,1600) (ICELG2(I,LCELL), I = 1, 10)
C
C        NEIGHBOUR CELLS AND THEIR POINTERS
C
         WRITE(JDEBUG,1700) LVSW, LCSW, LHSW, LHSE, LCSE, LVSE,
     1                      LVNE, LCNE, LHNE, LHNW, LCNW, LVNW
         IF (LVSW .NE. 0) THEN
            WRITE(JDEBUG,1800) (ICELG2(I,LVSW), I = 1, 10)
         ENDIF
         IF (LCSW .NE. 0) THEN
            WRITE(JDEBUG,1900) (ICELG2(I,LCSW), I = 1, 10)
         ENDIF
         IF (LHSW .NE. 0) THEN
            WRITE(JDEBUG,2000) (ICELG2(I,LHSW), I = 1, 10)
         ENDIF
C
         IF (LHSE .NE. 0) THEN
            WRITE(JDEBUG,2100) (ICELG2(I,LHSE), I = 1, 10)
         ENDIF
         IF (LCSE .NE. 0) THEN
            WRITE(JDEBUG,2200) (ICELG2(I,LCSE), I = 1, 10)
         ENDIF
         IF (LVSE .NE. 0) THEN
            WRITE(JDEBUG,2300) (ICELG2(I,LVSE), I = 1, 10)
         ENDIF
```

```
C
          IF (LVNE .NE. 0) THEN
            WRITE(JDEBUG,2400) (ICELG2(I,LVNE), I = 1, 10)
          ENDIF
          IF (LCNE .NE. 0) THEN
             WRITE(JDEBUG,2500) (ICELG2(I,LCNE), I = 1, 10)
          ENDIF
          IF (LHNE .NE. 0) THEN
             WRITE(JDEBUG,2600) (ICELG2(I,LHNE), I = 1, 10)
          ENDIF
C
          IF (LHNW .NE. 0) THEN
             WRITE(JDEBUG,2700) (ICELG2(I,LHNW), I = 1, 10)
          ENDIF
          IF (LCNW .NE. 0) THEN
             WRITE(JDEBUG,2800) (ICELG2(I,LCNW), I = 1, 10)
          ENDIF
          IF (LVNW .NE. 0) THEN
             WRITE(JDEBUG,2900) (ICELG2(I,LVNW), I = 1, 10)
          ENDIF
C
C         NEW CREATED CELLS
C
          WRITE(JDEBUG,4300) LMSW, LMSE, LMNE, LMNW
          WRITE(JDEBUG,4400) (ICELG2(I,LMSW),I=1,10),KAUXG2(LMSW)
          WRITE(JDEBUG,4500) (ICELG2(I,LMSE),I=1,10),KAUXG2(LMSE)
          WRITE(JDEBUG,4600) (ICELG2(I,LMNE),I=1,10),KAUXG2(LMNE)
          WRITE(JDEBUG,4700) (ICELG2(I,LMNW),I=1,10),KAUXG2(LMNW)

C         NEIGHBOURING CELLS OF ALL NODES OF LCELL
C
          WRITE(JDEBUG,3000) (NEIBG2(I,KSW),I=1,4)
          WRITE(JDEBUG,3100) (NEIBG2(I,KSE),I=1,4)
          WRITE(JDEBUG,3200) (NEIBG2(I,KNE),I=1,4)
          WRITE(JDEBUG,3300) (NEIBG2(I,KNW),I=1,4)
          WRITE(JDEBUG,3400) (NEIBG2(I,KS ),I=1,4)
          WRITE(JDEBUG,3500) (NEIBG2(I,KE ),I=1,4)
          WRITE(JDEBUG,3600) (NEIBG2(I,KN ),I=1,4)
          WRITE(JDEBUG,3700) (NEIBG2(I,KW ),I=1,4)
          WRITE(JDEBUG,4800) (NEIBG2(I,KC ),I=1,4)
C
C         BOUNDARY NODES
C
          IF (IGOTO .NE. 0) THEN
             WRITE(JDEBUG,3800) IONE, ICOR, ITWO
             WRITE(JDEBUG,3900) (IBNDG2(I,IONE),I=1,5)
             WRITE(JDEBUG,4000) (IBNDG2(I,ITWO),I=1,5)
             IF (ICOR .NE. 0) THEN
                WRITE(JDEBUG,4100) (IBNDG2(I,ICOR),I=1,5)
                ICOR = NBNDG2 - 1
                WRITE(JDEBUG,4900) ICOR,(IBNDG2(I,ICOR),I=1,5)
             ENDIF
             WRITE(JDEBUG,4900) NBNDG2,(IBNDG2(I,NBNDG2),I=1,5)
          ENDIF
        ENDIF       ! IWRITE
C
C       -----------------
```

```
C       FORMAT STATEMENTS
C       -----------------
C
1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM G2DIVO' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(/10X,'***** INFORMATION BEFORE DIVISION *****'/)
1400    FORMAT(5X,'NNODG2 =',I7,5X,'NCELG2 =',I7,5X,'NBNDG2 =',I7,
     1         5X,'LEVEL  =',I7                                    )
1500    FORMAT(5X,'CELL POINTERS FOR LCELL =',I6,/5X,'KC  =',I6,5X,
     1    'KSW =',I6,5X,'KS  =',I6,5X,'KSE =',I6,5X,'KE  =',I6/5X,
     2    'KNE =',I6,5X,'KN  =',I6,5X,'KNW =',I6,5X,'KW  =',I6,5X,
     3    'KX  =',Z7                                                )
1600    FORMAT(5X,'LCELL POINTERS',5X,10I6)
1700    FORMAT(5X,'CELLS NEIGHBOURING LCELL :'/
     1         5X,'LVSW=',I5,5X,'LCSW=',I5,5X,'LHSW=',I5,
     2         5X,'LHSE=',I5,5X,'LCSE=',I5,5X,'LVSE=',I5/
     3         5X,'LVNE=',I5,5X,'LCNE=',I5,5X,'LHNE=',I5,
     4         5X,'LHNW=',I5,5X,'LCNW=',I5,5X,'LVNW=',I5   )
1800    FORMAT(5X,'LVSW  POINTERS',5X,10I6)
1900    FORMAT(5X,'LCSW  POINTERS',5X,10I6)
2000    FORMAT(5X,'LHSW  POINTERS',5X,10I6)
2100    FORMAT(5X,'LHSE  POINTERS',5X,10I6)
2200    FORMAT(5X,'LCSE  POINTERS',5X,10I6)
2300    FORMAT(5X,'LVSE  POINTERS',5X,10I6)
2400    FORMAT(5X,'LVNE  POINTERS',5X,10I6)
2500    FORMAT(5X,'LCNE  POINTERS',5X,10I6)
2600    FORMAT(5X,'LHNE  POINTERS',5X,10I6)
2700    FORMAT(5X,'LHNW  POINTERS',5X,10I6)
2800    FORMAT(5X,'LCNW  POINTERS',5X,10I6)
2900    FORMAT(5X,'LVNW  POINTERS',5X,10I6)
3000    FORMAT(5X,'NEIGHBOUR CELLS OF KSW :',4I7)
3100    FORMAT(5X,'NEIGHBOUR CELLS OF KSE :',4I7)
3200    FORMAT(5X,'NEIGHBOUR CELLS OF KNE :',4I7)
3300    FORMAT(5X,'NEIGHBOUR CELLS OF KNW :',4I7)
3400    FORMAT(5X,'NEIGHBOUR CELLS OF KS  :',4I7)
3500    FORMAT(5X,'NEIGHBOUR CELLS OF KE  :',4I7)
3600    FORMAT(5X,'NEIGHBOUR CELLS OF KN  :',4I7)
3700    FORMAT(5X,'NEIGHBOUR CELLS OF KW  :',4I7)
3800    FORMAT(5X,'BOUNDARY NODE INFORMATION : '/
     1         5X,'IONE =',I6,5X,'ICOR =',I6,5X,'ITWO =',I6)
3900    FORMAT(5X,'B. POINTERS OF IONE :',5I6)
4000    FORMAT(5X,'B. POINTERS OF ITWO :',5I6)
4100    FORMAT(5X,'B. POINTERS OF ICOR :',5I6)
4200    FORMAT(//10X,'***** INFORMATION AFTER  DIVISION *****'/)
4300    FORMAT(5X,'NEW CREATED CELLS :'/
     1         5X,'LMSW=',I5,5X,'LMSE=',I5,5X,'LMNE=',I5,
     2         5X,'LMNW=',I5                                  )
4400    FORMAT(5X,'LMSW  POINTERS',5X,10I6,Z10)
4500    FORMAT(5X,'LMSE  POINTERS',5X,10I6,Z10)
4600    FORMAT(5X,'LMNE  POINTERS',5X,10I6,Z10)
4700    FORMAT(5X,'LMNW  POINTERS',5X,10I6,Z10)
4800    FORMAT(5X,'NEIGHBOUR CELLS OF KC  :',4I7)
4900    FORMAT(5X,'B. POINTERS OF ADDED NODE (',I6,')',5X,5I6)
C
        RETURN
        END
```

G2FROZ

```
      SUBROUTINE G2FROZ

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'HEXCOD.INC'

C***********************************************************************

C     THIS SUBROUTINE SCANS THE SPECIES CONCENTRATIONS FOR THE ROGERS
C     AND CHINITZ MODEL AND APPLIES CORRECTIVE PROCEDURE FOR FROZEN
C     CASE AND WHEN THE CONCENTRATION OF OH BECOMES VERY LARGE

C***********************************************************************
C
C
C     IF (KROGER .NE. 1    ) RETURN
C     IF (TRIGCH .GT. 1500.) RETURN
C
C     ATOMH2 = YMAXCH(3)*RAMWCH(3)
C     ATOMO2 = YMAXCH(1)*RAMWCH(1)
      YMAXOH = 0.5*YMAXCH(2)
      YMXDIF = 0.1*YMAXCH(2)
      YMXKNK = YMXDIF
      IF (IDBGG2 .EQ. 998) YMXDIF = 0.001*YMAXCH(2)
C
C     TOTAL NUMBER OF NORTHERN CELLS
      NORCEL = 0
C
C     TOTAL NUMBER OF NODES WHERE ADJUSTMENT MAY BE NEEDED
      NNODAD = 0
C
C     STEP THROUGH EACH CEWIC CELL
C
      DO 40 JCELL = 1, NCELA2
C
C       FIND THE ACTUAL CELL NUMBER
C
        ICELL = ICELA2(JCELL)
C
C       SET UP NODE POINTERS FOR THIS CELL
C
        KSW = ICELG2(2,ICELL)
        KSE = ICELG2(4,ICELL)
        KNE = ICELG2(6,ICELL)
        KNW = ICELG2(8,ICELL)
C
```

```
C         CHECK IF KSE IS AN INFLOW NODE
C
          IF (IAND(KAUXG2(ICELL),KLOOOF) .NE. 0) THEN
C
C             SCAN ALL BOUNDARY NODES
C
              DO 4 IBND = 1, NBNDG2
C
C                IS THE BOUNDARY NODE KSE
C
                 IF (IBNDG2(1,IBND) .EQ. KSE) THEN
C                    IS IT AN INFLOW NODE
                     IF (IBNDG2(5,IBND) .EQ. 2) KSW = KSE
                     GOTO 5
                 ENDIF
4             CONTINUE
C
C             IS THIS A NORTHERN CELL
C
5             IF (IAND(KAUXG2(ICELL),KLOOOC) .NE. 0) THEN
C
C                SCAN ALL BOUNDARY NODES
C
                 DO 7 IBND = 1, NBNDG2
C
C                   IS THE BOUNDARY NODE KNE
C
                    KNE = ICELG2(6,ICELL)

                    IF (IBNDG2(1,IBND) .EQ. KNE) THEN
C                       IS IT AN INFLOW NODE, IF NOT KEEP FOR MORE ...
                        IF (IBNDG2(5,IBND) .NE. 2) THEN
                            NORCEL = NORCEL + 1
                            MRKDA2(NORCEL) = ICELL
                            NNODAD         = NNODAD + 1
                            MRKCA2(NNODAD) = KNE
                        ENDIF
                        GOTO 8
                    ENDIF

7                CONTINUE

              ENDIF
8             IF (KSE .EQ. KSW) GOTO 40

          ENDIF

          NNODAD         = NNODAD + 1
          MRKCA2(NNODAD) = KSE
C
C
C         FOR PRE-MIXED FLOWS ALSO APPLY ATOM CONSERVATION EQUATIONS
C
C          IF (IALOCH(6,3) .EQ. -9) THEN
C
C              YO2SE  = DPENG2(5,KSE)/DPENG2(1,KSE)
C              YH2SE  = DPENG2(7,KSE)/DPENG2(1,KSE)
```

```
C              COHSE  = 2.*( 2.*(ATOMO2-YO2SE*RAMWCH(1))
C     1                      - (ATOMH2-YH2SE*RAMWCH(3)))
C
C              IF (COHSE .LT. 0.) THEN
C                 DPENG2(5,KSE) = 1.01*DPENG2(5,KSE)
C                 DPENG2(6,KSE) = 0.
C              ELSE
C                 DPENG2(6,KSE) = AMWTCH(2)*COHSE*DPENG2(1,KSE)
C              ENDIF
C
C           ENDIF
C
C           STORE OH DENSITY AT TWO NODES
C
            YOHSW  = DPENG2(6,KSW)/DPENG2(1,KSW)
            YOHSE  = DPENG2(6,KSE)/DPENG2(1,KSE)
C
C           CHECK IF YOH IS CLOSE TO MAXIMUM POSSIBLE VALUE
C           THIS MUST NOT REALLY BE POSSIBLE SINCE SOME OF THE SPECIES
C           MUST GET CONSUMED TO PRODUCE H2O
C
            IF (YOHSW .GT. YMAXOH) THEN
               DO 10 IQ = 5, NEQNFL
                  YSPSW          = DPENG2(IQ,KSW)/DPENG2(1,KSW)
                  YSPSE          = DPENG2(IQ,KSE)/DPENG2(1,KSE)
                  YSPSE          = 0.5*(YSPSE+YSPSW)
                  DPENG2(IQ,KSE) = YSPSE*DPENG2(1,KSE)
10             CONTINUE
               GOTO 40
            ENDIF
C
C           CHECK IF DIFFERENCE OF YOH IS LARGE NEAR THE TWO NODES
C           THIS IS AN ATTEMPT TO AVOID SUDDEN JUMPS
C
            CHECKY = ABS(YOHSE-YOHSW)
C
            IF (CHECKY .GT. YMXKNK) THEN
               DO 20 IQ = 5, NEQNFL
                  YSPSW          = DPENG2(IQ,KSW)/DPENG2(1,KSW)
                  YSPSE          = DPENG2(IQ,KSE)/DPENG2(1,KSE)
                  YSPSE          = 0.5*(YSPSE+YSPSW)
                  DPENG2(IQ,KSE) = YSPSE*DPENG2(1,KSE)
20             CONTINUE
               GOTO 40
            ENDIF
C
C           CHECK IF THERE IS FROZEN FLOW; IF SO SIMPLY CONVECT THE VALUES
C
            TEMP = TEMPG2(KSE)*TREFFL
C
            IF (TEMP .GT. TRIGCH) GOTO 40
C
            IF (CHECKY .GT. YMXDIF) THEN
               DO 30 IQ = 5, NEQNFL
                  YSPSW          = DPENG2(IQ,KSW)/DPENG2(1,KSW)
                  DPENG2(IQ,KSE) = YSPSW*DPENG2(1,KSE)
30             CONTINUE
```

```
          ENDIF
C
40      CONTINUE
C
C       REPEAT THE WHOLE PROCESS FOR NORTHERN CELLS
C
        DO 80 JCELL = 1, NORCEL
C
C         FIND THE ACTUAL CELL NUMBER
C
          ICELL = MRKDA2(JCELL)
C
C         SET UP NODE POINTERS FOR THIS CELL
C
          KNE = ICELG2(6,ICELL)
          KNW = ICELG2(8,ICELL)

C         FOR PRE-MIXED FLOWS ALSO APPLY ATOM CONSERVATION EQUATIONS
C
C          IF (IALOCH(6,3) .EQ. -9) THEN
C
C              YO2NE  = DPENG2(5,KNE)/DPENG2(1,KNE)
C              YH2NE  = DPENG2(7,KNE)/DPENG2(1,KNE)
C              COHNE  = 2.*( 2.*(ATOMO2-YO2NE*RAMWCH(1))
C     1                       - (ATOMH2-YH2NE*RAMWCH(3)))
C
C              IF (COHNE .LT. 0.) THEN
C                 DPENG2(5,KNE) = 1.01*DPENG2(5,KNE)
C                 DPENG2(6,KNE) = 0.
C              ELSE
C                 DPENG2(6,KNE) = AMWTCH(2)*COHNE*DPENG2(1,KNE)
C              ENDIF
C
C           ENDIF
C
C         STORE OH DENSITY AT TWO NODES
C
          YOHNE  = DPENG2(6,KNE)/DPENG2(1,KNE)
          YOHNW  = DPENG2(6,KNW)/DPENG2(1,KNW)
C
C         CHECK IF YOH IS CLOSE TO MAXIMUM POSSIBLE VALUE
C         THIS MUST NOT REALLY BE POSSIBLE SINCE SOME OF THE SPECIES
C         MUST GET CONSUMED TO PRODUCE H2O
C
          CHECKY = YMAXCH(2) - YOHNE
C
          IF (YOHSW .GT. YMAXOH) THEN
             DO 50 IQ = 5, NEQNFL
                YSPNE          = DPENG2(IQ,KNE)/DPENG2(1,KNE)
                YSPNW          = DPENG2(IQ,KNW)/DPENG2(1,KNW)
                YSPNE          = 0.5*(YSPNW+YSPNE)
                DPENG2(IQ,KNE) = YSPNE*DPENG2(1,KNE)
50           CONTINUE
             GOTO 80
          ENDIF
C
C         CHECK IF DIFFERENCE OF YOH IS LARGE NEAR THE TWO NODES
```

```
C         THIS IS AN ATTEMPT TO AVOID SUDDEN JUMPS
C
          CHECKY = ABS(YOHNW-YOHNE)
C
          IF (CHECKY .GT. YMXKNK) THEN
             DO 60 IQ = 5, NEQNFL
                YSPNE          = DPENG2(IQ,KNE)/DPENG2(1,KNE)
                YSPNW          = DPENG2(IQ,KNW)/DPENG2(1,KNW)
                YSPNE          = 0.5*(YSPNW+YSPNE)
                DPENG2(IQ,KNE) = YSPNE*DPENG2(1,KNE)
60           CONTINUE
             GOTO 80
          ENDIF
C
C         CHECK IF THERE IS FROZEN FLOW; IF SO SIMPLY CONVECT THE VALUES
C
          TEMP = TEMPG2(KNE)*TREFFL
C
          IF (TEMP .GT. TRIGCH) GOTO 80
C
          IF (CHECKY .GT. YMXDIF) THEN
             DO 70 IQ = 5, NEQNFL
                YSPNW          = DPENG2(IQ,KNW)/DPENG2(1,KNW)
                DPENG2(IQ,KNE) = YSPNW*DPENG2(1,KNE)
70           CONTINUE
          ENDIF
C
80      CONTINUE
C
        IF (NEQNFL .EQ. 8) RETURN
C        IF (IALOCH(6,3) .NE. -9) THEN
C           DO 99 INODE = 1, NNODG2
CC
C              RHORPR = DPENG2(1,INODE)
C              YO2    = DPENG2(5,INODE)/RHORPR
C              YH2    = DPENG2(7,INODE)/RHORPR
C
C              IF (NEQNFL .EQ. 8) THEN
C                 CH2O  = 2.*( -(ATOMO2-YO2*RAMWCH(1))
C     1                        +(ATOMH2-YH2*RAMWCH(3)))
C                 IF (CH2O .LT. 0.) THEN
C                    DPENG2(8,INODE) = 0.
C                 ELSE
C                    DPENG2(8,INODE) = AMWTCH(4)*CH2O*DPENG2(1,INODE)
C                 ENDIF
C              ENDIF
C99         CONTINUE
C           RETURN
C        ENDIF
C
C       SCAN ALL THE INTERIOR NODES FOR THE ROGERS AND CHINITZ MODEL
C       WHERE THE CONCENTRATION OF H2O IS NEGATIVE
C       RESET THE DEPENDENT VARIABLES IF NEED BE
C
        DO 100 JNODE = 1, NNODAD

           INODE  = MRKCA2(JNODE)
```

```
C
      RHORPR = DPENG2(1,INODE)
      YO2    = DPENG2(5,INODE)/RHORPR
      YOH    = DPENG2(6,INODE)/RHORPR
      YH2    = DPENG2(7,INODE)/RHORPR
      YH2O   = DPENG2(8,INODE)/RHORPR
      YH2O   = 1.-YO2-YOH-YH2-YNRTCH
C
      IF (YH2O .GT. 0.) GOTO 100
C
C     2H2 + O2 == 2H2O
C
      CONO2  = YO2 *RAMWCH(1)
      CONH2  = YH2 *RAMWCH(3)
      CONH2O =-YH2O*RAMWCH(4)
      XXX    = MIN (0.5*CONH2,CONO2,0.5*CONH2O)
C
      YO2    = AMWTCH(1)*(CONO2-XXX)
      YH2    = AMWTCH(3)*(CONH2-2.*XXX)
      YH2O   = AMWTCH(4)*(2.*XXX-CONH2O)
C
      IF (YH2O .GT. 0.) THEN
          DPENG2(5,INODE) = RHORPR*YO2
          DPENG2(7,INODE) = RHORPR*YH2
          GOTO 100
      ENDIF
C
C     H2 + 2OH == 2H2O
C
      IF (YH2 .LE. 0.)  GOTO 90
C
      CONH2  = YH2 *RAMWCH(3)
      CONOH  = YOH *RAMWCH(2)
      CONH2O =-YH2O*RAMWCH(4)
      XXX    = MIN (CONH2,0.5*CONOH,0.5*CONH2O)
C
      YH2    = AMWTCH(3)*(CONH2-XXX)
      YOH    = AMWTCH(2)*(CONOH-2.*XXX)
      YH2O   = AMWTCH(4)*(2.*XXX-CONH2O)
C
      IF (YH2O .GT. 0.) THEN
          DPENG2(5,INODE) = RHORPR*YO2
          DPENG2(6,INODE) = RHORPR*YOH
          DPENG2(7,INODE) = RHORPR*YH2
          GOTO 100
      ENDIF
C
C     4OH == 2H2O + O2
C
90    CONO2  = YO2 *RAMWCH(1)
      CONOH  = YOH *RAMWCH(2)
      CONH2O =-YH2O*RAMWCH(4)
      XXX    = MIN (0.25*CONOH,0.5*CONH2O)
C
      YOH    = AMWTCH(2)*(CONOH-4.*XXX)
      YO2    = AMWTCH(1)*(CONO2+XXX)
      YH2O   = AMWTCH(4)*(2.*XXX-CONH2O)
```

```
C
         DPENG2(5,INODE) = RHORPR*YO2
         DPENG2(6,INODE) = RHORPR*YOH
         DPENG2(7,INODE) = RHORPR*YH2
C
C        IF (YH2O .LT. 0.) write(6,*) ' g2froz  yh2o still neg',yh2o
C
100     CONTINUE
C
        RETURN
        END
```

G2HANG

```
        SUBROUTINE G2HANG
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'TICOMN.INC'
        INCLUDE 'HEXCOD.INC'
        DIMENSION MARK(0:MNODG2)
C
C*****************************************************************
C
C       THIS SUBROUTINE COMPUTES THE NODES AT VARIOUS TEMPORAL LEVEL
C       CELLS.  THIS MAKES THE UPDATING AND CONVERGENCE HISTORY
C       COLLECTION A LITTLE EASIER AND EFFICIENT.
C
C*****************************************************************
C
C       MARK ALL THE NODES FOR SUBSEQUENT COLLECTION
C
        MARK(0) = 0
        DO 10 INODE = 1, NNODG2
           MARK(INODE) = 1
10      CONTINUE
C
C       TOTAL NUMBER OF HANGING NODES
        NHNGA2 = 0
C
C       TOTAL NUMBER OF "NORMAL" NODES
        NNODA2 = 0
C
C       LOOP OVER ALL THE TEMPORAL LEVELS TO COLLECT NODES
C
        DO 30 ITGL = 0, NMAXTI
C
C           COLLECT THE FIRST NODE AT THIS LEVEL
            ILVLA2(1,ITGL) = NNODA2 + 1
C
C           LOOP OVER ALL THE CELLS AT THIS LEVEL AND CLASSIFY NODES
```

```
C           ACCORDING TO THE TEMPORAL LEVEL

            DO 20 JCELL = ILVLTI(1,ITGL), ILVLTI(2,ITGL)
C
C              NODE/CELL ASSIGNMENTS
C
               ICELL = ICELTI (  JCELL)
               KSW   = ICELG2 (2,ICELL)
               KS    = ICELG2 (3,ICELL)
               KSE   = ICELG2 (4,ICELL)
               KE    = ICELG2 (5,ICELL)
               KNE   = ICELG2 (6,ICELL)
               KN    = ICELG2 (7,ICELL)
               KNW   = ICELG2 (8,ICELL)
               KW    = ICELG2 (9,ICELL)
               KXB1  = IAND (KAUXG2(ICELL),KL0009)
C
C              CHECK IS THE CELL HAS NODES WITH FIXED BOUNDARY CONDITIONS
C
               IF ( KXB1 .EQ. 9 ) THEN
                   MARK(KNW) = 0
                   MARK(KW ) = 0
                   MARK(KSW) = 0
               ENDIF
C
C              CHECK IF COLLECTION IS NEEDED AT THE SOUTHWESTERN NODE
C
               IF ( MARK(KSW) .NE. 0 ) THEN
                   MARK(KSW)      = 0
                   NNODA2         = NNODA2 + 1
                   MRKDA2(NNODA2) = KSW
               ENDIF
C
C              CHECK IF COLLECTION IS NEEDED AT THE SOUTHEASTERN NODE
C
               IF ( MARK(KSE) .NE. 0 ) THEN
                   MARK(KSE)      = 0
                   NNODA2         = NNODA2 + 1
                   MRKDA2(NNODA2) = KSE
               ENDIF
C
C              CHECK IF COLLECTION IS NEEDED AT THE NORTHEASTERN NODE
C
               IF ( MARK(KNE) .NE. 0 ) THEN
                   MARK(KNE)      = 0
                   NNODA2         = NNODA2 + 1
                   MRKDA2(NNODA2) = KNE
               ENDIF
C
C              CHECK IF COLLECTION IS NEEDED AT THE NORTHWESTERN NODE
C
               IF ( MARK(KNW) .NE. 0 ) THEN
                   MARK(KNW)      = 0
                   NNODA2         = NNODA2 + 1
                   MRKDA2(NNODA2) = KNW
               ENDIF
C
```

```
C              CHECK IF SOUTHERN NODE IS A HANGING NODE
C
               IF ( MARK(KS) .NE. 0 ) THEN
                   MARK(KS)       = 0
                   NHNGA2         = NHNGA2 + 1
                   MRKCA2(NHNGA2) = KS
                   WORKA2(NHNGA2) = KSW
                   CHNGA2(NHNGA2) = KSE
               ENDIF
C
C              CHECK IF EASTERN NODE IS A HANGING NODE
C
               IF ( MARK(KE) .NE. 0 ) THEN
                   MARK(KE)       = 0
                   NHNGA2         = NHNGA2 + 1
                   MRKCA2(NHNGA2) = KE
                   WORKA2(NHNGA2) = KSE
                   CHNGA2(NHNGA2) = KNE
               ENDIF
C
C              CHECK IF NORTHERN NODE IS A HANGING NODE
C
               IF ( MARK(KN) .NE. 0 ) THEN
                   MARK(KN)       = 0
                   NHNGA2         = NHNGA2 + 1
                   MRKCA2(NHNGA2) = KN
                   WORKA2(NHNGA2) = KNE
                   CHNGA2(NHNGA2) = KNW
               ENDIF
C
C              CHECK IF WESTERN NODE IS A HANGING NODE
C
               IF ( MARK(KW) .NE. 0 ) THEN
                   MARK(KW)       = 0
                   NHNGA2         = NHNGA2 + 1
                   MRKCA2(NHNGA2) = KW
                   WORKA2(NHNGA2) = KNW
                   CHNGA2(NHNGA2) = KSW
               ENDIF
C
20          CONTINUE
C
C           COLLECT THE LAST NODE AT THIS LEVEL
            ILVLA2(2,ITGL) = NNODA2
C
30      CONTINUE
C
        RETURN
        END
```

G2IBLC

```
        SUBROUTINE G2IBLC
```

```
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'

C*********************************************************************
C
C     THIS SUBROUTINE SETS UP THE CELL, BOUNDARY CONDITION,  AND
C     MULTIPLE-GRID-LEVEL ARRAYS FOR THE GLOBAL MESHES WHICH WERE
C     GENERATED BY THE BLOCK GRID GENERATOR
C
C*********************************************************************
C
      MALVG2 = IPASKY(23)
      IDBGG2 = IPASKY(37)
C
C     READ EVERYTHING FROM A PREVIOUSLY WRITTEN FILE
C
C     INTEGERS FROM G2COMN.INC

      READ (JREADG,1100) NNODG2, NCELG2, NBNDG2
      DO 10 LC = 1, NCELG2
        READ (JREADG,1100) (ICELG2(IP,LC), IP=1,10), KAUXG2(LC)
10    CONTINUE

      DO 20 IB = 1, NBNDG2
        READ (JREADG,1100) (IBNDG2(IP,IB), IP = 1, 5)
20    CONTINUE

      DO 30 IN = 1, NNODG2
        READ (JREADG,1100) (NEIBG2(IP,IN), IP = 1, 4)
30    CONTINUE

      DO 40 LV = -MLVLG2, MLVLG2
        READ (JREADG,1100) (ILVLG2(IP,LV), IP = 1, 3)
40    CONTINUE

      READ (JREADG,1100) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

      DO 50 IN = 1, NNODG2
        READ(JREADG,1200) GEOMG2(1,IN),GEOMG2(2,IN)
50    CONTINUE
C
C     -----------------
C     FORMAT STATEMENTS
C     -----------------

1100  FORMAT(11I7)
1200  FORMAT(8E15.6)
C
      RETURN
      END
```

G2IBOG

```
      SUBROUTINE G2IBOG (NXRECT, NYRECT, XSOUTH,XEAST,XNORTH,XWEST,
     1                   YSOUTH,YEAST,YNORTH,YWEST,GEOMGG)

      DIMENSION XEAST (*),   XSOUTH(*),   XWEST (*),   XNORTH(*),
     1          YEAST (*),   YSOUTH(*),   YWEST (*),   YNORTH(*),
     2          GEOMGG(2,*)
      DIMENSION DISTW(1000),   DISTE(1000)

C*********************************************************************
C
C       THIS SUBROUTINE IS IN-BOUNDARY-OUT-GRID (IBOG); I.E., IT TAKES
C       IN THE BOUNDARY INFORMATION AND GENERATES THE INTERIOR GRID.
C       THIS MAY BE JUST ONE OF THE SECTIONS OF THE OVERALL GRID.  EACH
C       SECTION IS GRIDDED BY AN ALGEBRAIC CONSTRUCTION.  NXRECT, NYRECT
C       CONTAIN THE NUMBER OF NODES ALONG EACH COORDINATE DIRECTION.
C       THE BOUNDARY INFORMATION IS CONTAINED IN XSOUTH, ..., YWEST.
C       THE OUTPUT IS THE TWO-DIMENSIONAL ARRAY GEOMGG THAT CONTAINS
C       THE (X,Y) COORDINATES OF THE DOMAIN.
C
C*********************************************************************
C
C       COMPUTE THE NODE BEFORE THE FIRST NORTH ONE (L IN FIG.)
C       AND THE MAXIMUM NUMBER OF NODES

        NBEFNO = NXRECT*(NYRECT-1)
        NNODG2 = NXRECT* NYRECT
C
C       SET SOUTH AND NORTH NODE INFORMATION

        DO 10 IX = 1, NXRECT
           NOS           = IX
           NON           = IX + NBEFNO
           GEOMGG(1,NOS) = XSOUTH(IX)
           GEOMGG(2,NOS) = YSOUTH(IX)
           GEOMGG(1,NON) = XNORTH(IX)
           GEOMGG(2,NON) = YNORTH(IX)
10      CONTINUE
C
C       SET WEST AND EAST NODE INFORMATION

        DO 20 IY = 1, NYRECT
           NOW           = 1 + (IY-1)*NXRECT
           NOE           = IY*NXRECT
           GEOMGG(1,NOW) = XWEST (IY)
           GEOMGG(2,NOW) = YWEST (IY)
           GEOMGG(1,NOE) = XEAST (IY)
           GEOMGG(2,NOE) = YEAST (IY)
20      CONTINUE
C
C       INITIALIZE THE FRACTIONAL DISTANCES ON WEST AND EAST EDGES
C
        DISTW(1) = 0.
        DISTE(1) = 0.
C
```

```
C       CALCULATE THE TOTAL DISTANCES ON WEST AND EAST EDGES
C
        DO 30 J = 2, NYRECT
           JM1      = J - 1
C
           INDJW    = 1 + (J  -1)*NXRECT
           INDJMW   = 1 + (JM1-1)*NXRECT
           INDJE    = J  *NXRECT
           INDJME   = JM1*NXRECT
C
           DXW      = GEOMGG(1,INDJW) - GEOMGG(1,INDJMW)
           DYW      = GEOMGG(2,INDJW) - GEOMGG(2,INDJMW)
           DXE      = GEOMGG(1,INDJE) - GEOMGG(1,INDJME)
           DYE      = GEOMGG(2,INDJE) - GEOMGG(2,INDJME)
C
           DISTW(J) = DISTW(JM1) + SQRT(DXW*DXW + DYW*DYW)
           DISTE(J) = DISTE(JM1) + SQRT(DXE*DXE + DYE*DYE)
C
30      CONTINUE

C       CALCULATE THE FRACTIONAL DISTANCES ON WEST AND EAST EDGES
C       FOR EACH NODE

        DO 40 J = 2, NYRECT
           DISTW(J) = DISTW(J)/DISTW(NYRECT)
           DISTE(J) = DISTE(J)/DISTE(NYRECT)
40      CONTINUE
C
C       STEP THROUGH EACH INTERIOR LINE
C
        DO 60 I = 2, NXRECT-1
           FRACI = FLOAT(I-1)/FLOAT(NXRECT-1)
C
C          CALCULATE FRACTIONAL DISTANCES FOR EACH INTERIOR POINT
C
           DO 50 J = 2,NYRECT-1
              FRACJ         = (1.-FRACI)*DISTW(J) + FRACI*DISTE(J)
C
              IND           = I + (     J-1)*NXRECT
              INDN          = I + (NYRECT-1)*NXRECT
              INDS          = I
C
C             COMPUTE THE DISTANCE FROM NORTH EDGE TO SOUTH EDGE
C
              DELXNS        = GEOMGG(1,INDN) - GEOMGG(1,INDS)
              DELYNS        = GEOMGG(2,INDN) - GEOMGG(2,INDS)
C
C             COMPUTE LOCATION OF INTERIOR POINT
C
              GEOMGG(1,IND) = GEOMGG(1,INDS) + FRACJ*DELXNS
              GEOMGG(2,IND) = GEOMGG(2,INDS) + FRACJ*DELYNS
C
50         CONTINUE
60      CONTINUE
C
C          ------------
C          NOMENCLATURE
```

```
C           ------------
C           -                                                  NX=NXRECT
C           -                                                  NY=NYRECT
C                          L  L  L                             L =NBEFNO
C                          +  +  + . . .
C                          1  2  3
C            1+(NY-1)*NX +--+--+--+--+--+--+ NY*NX
C                          +       NORTH      + (NY-1)*NX = L
C                          + W                E +
C  INDJW                   + E                A +                  INDJE
C                     ...  + S                S + ...
C                  1+2*NX + T                T + 3*NX
C                    1+NX +       SOUTH        + 2*NX
C                       1 +--+--+--+--+--+--+ NX
C                            2  3  ...     NX-1
C
C
        RETURN
        END
```

G2INIT

```
        SUBROUTINE G2INIT

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'HEXCOD.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'KYCOMN.INC'

C*********************************************************************
C
C       THIS SUBROUTINE SETS UP THE CELL, BOUNDARY CONDITION,  AND
C       MULTIPLE-GRID-LEVEL ARRAYS FOR THE GLOBAL MESHES
C
C*********************************************************************
C
C           ------------
C           NOMENCLATURE
C           ------------
C                                                              NX=NXRECT
C                                                              NY=NYRECT
C                          L  L  L                             L =NBEFNO
C                          +  +  + . . .
C                          1  2  3
C            1+(NY-1)*NX +--+--+--+--+--+--+ NY*NX
C                          +       NORTH      + (NY-1)*NX = L
C                          + W                E +
C  INDJW                   + E                A +                  INDJE
C                     ...  + S                S + ...
C                  1+2*NX + T                T + 3*NX
C                    1+NX +       SOUTH        + 2*NX
```

```
C                    1 +--+--+--+--+--+--+--+ NX
C          -              2  3  ...      NX-1
C          -
C
C
C
        MALVG2 = IPASKY(23)
        IDBGG2 = IPASKY(37)

C       READ ALL THE INFORMATION FROM INPUTG.DAT

        READ (JREADG,1000) NXRECT, NYRECT, NBNDG2, NNODG2
        READ (JREADG,1000) (IBNDG2(5,IB), IB=1,NBNDG2)
        READ (JREADG,1100) (GEOMG2(1,KN),GEOMG2(2,KN), KN=1,NNODG2)

C       CHECK FOR OVERFLOW IN BOUNDARY NODE ARRAYS

        IF(NBNDG2 .GT. MBNDG2) THEN
          ZER1 = NBNDG2
          ZER2 = MBNDG2
          CALL ERRORM (8,'G2INIT','NBNDG2',ZER1,'MBNDG2',ZER2,JPRINT,
     1      'NUMBER OF BOUNDARY NODES EXCEEDS ITS LIMIT')
        ENDIF

        IF(NNODG2 .GT. MNODG2) THEN
          ZER1 = NNODG2
          ZER2 = MNODG2
          CALL ERRORM (6,'G2INIT','NNODG2',ZER1,'MNODG2',ZER2,JPRINT,
     1      'NUMBER OF NODES EXCEEDS ITS LIMIT')
        ENDIF
C
C       COMPUTE NUMBER OF CELLS IN EACH DIRECTION ON THE GLOBAL MESH
C
        NXCELL = NXRECT - 1
        NYCELL = NYRECT - 1
C
C       INITIALIZE THE NUMBER OF CELLS AND BOUNDARY CONDITION POINTERS
C
        NCELG2 = 0
        NBNDG2 = 0
C
C       INITIALIZE POINTERS FOR ALL LEVELS
C
        DO 30 ILEVEL = -MLVLG2, MLVLG2
           ILVLG2(1,ILEVEL) = 0
           ILVLG2(2,ILEVEL) = 0
           ILVLG2(3,ILEVEL) = 0
30      CONTINUE
C
C       LOOP THROUGH ALL COARSER GRID LEVELS (IF ANY)
C
        ISTART = MIN (NCRSG2, MLVLG2-1)

        DO 50 ICOARS = ISTART, 1, -1
          ISIZE = 2**ICOARS
C
C         LOOP THROUGH EACH CELL ON THIS LEVEL
C
```

```
      DO 40 JCELL = 1, NYCELL, ISIZE
      DO 40 ICELL = 1, NXCELL, ISIZE
C
        NCELG2 = NCELG2 + 1
C
C       FIND THE CENTER OF THIS CELL
C
        ICELG2(1,NCELG2) = (ICELL+ISIZE/2)+(JCELL+ISIZE/2-1)*NXRECT
C
C       COMPUTE INDICES OF ALL BOUNDING NODES
C
        ICELG2(2 ,NCELG2) = (ICELL       )+(JCELL       -1)*NXRECT
        ICELG2(3 ,NCELG2) = (ICELL+ISIZE/2)+(JCELL       -1)*NXRECT
        ICELG2(4 ,NCELG2) = (ICELL+ISIZE  )+(JCELL       -1)*NXRECT
        ICELG2(5 ,NCELG2) = (ICELL+ISIZE  )+(JCELL+ISIZE/2-1)*NXRECT
        ICELG2(6 ,NCELG2) = (ICELL+ISIZE  )+(JCELL+ISIZE  -1)*NXRECT
        ICELG2(7 ,NCELG2) = (ICELL+ISIZE/2)+(JCELL+ISIZE  -1)*NXRECT
        ICELG2(8 ,NCELG2) = (ICELL       )+(JCELL+ISIZE  -1)*NXRECT
        ICELG2(9 ,NCELG2) = (ICELL       )+(JCELL+ISIZE/2-1)*NXRECT
        ICELG2(10,NCELG2) = 0
C
C       INITIALIZE AUXILIARY CELL INFORMATION
C
        KAUXG2(NCELG2) = 0
C
40    CONTINUE
C
C     SET UP THE MULTIPLE-GRID-LEVEL ARRAY FOR THIS LEVEL
C
      ILEVEL           = -ICOARS
      ILVLG2(1,ILEVEL) = ILVLG2(2,ILEVEL-1) + 1
      ILVLG2(2,ILEVEL) = NCELG2
      ILVLG2(3,ILEVEL) = ILVLG2(2,ILEVEL) - ILVLG2(1,ILEVEL) + 1
C
C     GO BACK FOR A FINER COARSE LEVEL
C
50  CONTINUE
C
C   LOOP THROUGH EACH GLOBAL CELL
C
    DO 60 JCELL = 1, NYCELL
      DO 60 ICELL = 1, NXCELL
C
        NCELG2 = NCELG2 + 1
C
C       COMPUTE INDICES OF CORNER OF CELL
C
        ICELG2(2,NCELG2) = ICELL     + (JCELL-1)*NXRECT
        ICELG2(4,NCELG2) = ICELL + 1 + (JCELL-1)*NXRECT
        ICELG2(6,NCELG2) = ICELL + 1 + (JCELL  )*NXRECT
        ICELG2(8,NCELG2) = ICELL     + (JCELL  )*NXRECT
C
C       INITIALLY, THERE IS NO NODE IN THE CENTER OF A FINE CELL
C
        ICELG2(1,NCELG2) = 0
C
C       THERE ARE NO NODES IN THE CENTER OF THE SIDES OF A FINE CELL
```

```
C
          ICELG2(3 ,NCELG2) = 0
          ICELG2(5 ,NCELG2) = 0
          ICELG2(7 ,NCELG2) = 0
          ICELG2(9 ,NCELG2) = 0
          ICELG2(10,NCELG2) = 0
C
C         INITIALIZE AUXILIARY CELL INFORMATION
C
          KAUXG2(NCELG2) = 0
C
60      CONTINUE
C
C       SET UP THE MULTIPLE-GRID-LEVEL ARRAY FOR THE GLOBAL FINE LEVEL
C
        ILVLG2(1,0) = ILVLG2(2,-1) + 1
        ILVLG2(2,0) = NCELG2
        ILVLG2(3,0) = ILVLG2(2,0) - ILVLG2(1,0) + 1
C
C       INITIALIZE THE MULTIPLE-GRID-LEVEL ARRAY FOR ALL EMBEDDED MESHES
C
        DO 70 ILEVEL = 1, MLVLG2
          ILVLG2(1,ILEVEL) = NCELG2 + 1
          ILVLG2(2,ILEVEL) = NCELG2
          ILVLG2(3,ILEVEL) = 0
70      CONTINUE
C
C       SET UP THE BOUNDARY CONDITION ARRAYS AND SET BOUNDARY POINTERS FOR
C       BOUNDARY NODES...
C
C       SOUTHWESTERN CORNER
C
        NBNDG2                    = NBNDG2 + 1
        IBNDG2(1,NBNDG2)          = 1
        IBNDG2(2,NBNDG2)          = ILVLG2(2,-1) + 1
        IBNDG2(3,NBNDG2)          = 0
        IBNDG2(4,NBNDG2)          = 2
        KAUXG2(IBNDG2(2,NBNDG2)) = IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000B)
        NBCPG2(1,2)               = NBNDG2 + 1
C
C       SOUTHERN EDGE
C
        DO 80 IBOUND = 2, NXRECT - 1
          NBNDG2                   = NBNDG2 + 1
          IBNDG2(1,NBNDG2)         = IBOUND
          IBNDG2(2,NBNDG2)         = ILVLG2(2,-1) + IBOUND - 1
          IBNDG2(3,NBNDG2)         = ILVLG2(2,-1) + IBOUND
          IBNDG2(4,NBNDG2)         = 3
          KAUXG2(IBNDG2(2,NBNDG2))= IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0003)
          KAUXG2(IBNDG2(3,NBNDG2))= IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL0003)
80      CONTINUE
C
C       SOUTHEASTERN CORNER
C
        NBNDG2                    = NBNDG2 + 1
        IBNDG2(1,NBNDG2)          = NXRECT
        IBNDG2(2,NBNDG2)          = ILVLG2(2,-1) + NXCELL
```

```
      IBNDG2(3,NBNDG2)          = 0
      IBNDG2(4,NBNDG2)          = 4
      KAUXG2(IBNDG2(2,NBNDG2)) = IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0007)
      NBCPG2(2,1)               = NBNDG2 - 1
      NBCPG2(2,2)               = NBNDG2 + 1
C
C     EASTERN EDGE
C
      DO 90 IBOUND = 2, NYRECT - 1
        NBNDG2                   = NBNDG2 + 1
        IBNDG2(1,NBNDG2)         = IBOUND*NXRECT
        IBNDG2(2,NBNDG2)         = ILVLG2(2,-1) + (IBOUND-1)*NXCELL
        IBNDG2(3,NBNDG2)         = ILVLG2(2,-1) + IBOUND*NXCELL
        IBNDG2(4,NBNDG2)         = 5
        KAUXG2(IBNDG2(2,NBNDG2))= IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0006)
        KAUXG2(IBNDG2(3,NBNDG2))= IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL0006)
90    CONTINUE
C
C     NORTHEASTERN CORNER
C
      NBNDG2                    = NBNDG2 + 1
      IBNDG2(1,NBNDG2)          = NXRECT*NYRECT
      IBNDG2(2,NBNDG2)          = ILVLG2(2,-1) + NXCELL*NYCELL
      IBNDG2(3,NBNDG2)          = 0
      IBNDG2(4,NBNDG2)          = 6
      KAUXG2(IBNDG2(2,NBNDG2)) = IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000E)
      NBCPG2(3,1)               = NBNDG2 - 1
      NBCPG2(3,2)               = NBNDG2 + 1
C
C     NORTHERN EDGE
C
      DO 100 IBOUND = NXRECT-1, 2, -1
        NBNDG2           = NBNDG2 + 1
        IBNDG2(1,NBNDG2) = NXRECT*(NYRECT-1) + IBOUND
        IBNDG2(2,NBNDG2) = ILVLG2(2,-1) + NXCELL*(NYCELL-1)+IBOUND
        IBNDG2(3,NBNDG2) = ILVLG2(2,-1) + NXCELL*(NYCELL-1)+IBOUND-1
        IBNDG2(4,NBNDG2) = 7
        KAUXG2(IBNDG2(2,NBNDG2))= IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000C)
        KAUXG2(IBNDG2(3,NBNDG2))= IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL000C)
100   CONTINUE
C
C     NORTHWESTERN CORNER
C
      NBNDG2                    = NBNDG2 + 1
      IBNDG2(1,NBNDG2)          = NXRECT*(NYRECT-1) + 1
      IBNDG2(2,NBNDG2)          = ILVLG2(2,-1) + NXCELL*(NYCELL-1) + 1
      IBNDG2(3,NBNDG2)          = 0
      IBNDG2(4,NBNDG2)          = 8
      KAUXG2(IBNDG2(2,NBNDG2)) = IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL000D)
      NBCPG2(4,1)               = NBNDG2 - 1
      NBCPG2(4,2)               = NBNDG2 + 1
C
C     WESTERN EDGE
C
      DO 110 IBOUND = NYRECT-1, 2, -1
        NBNDG2                   = NBNDG2 + 1
        IBNDG2(1,NBNDG2)         = NXRECT*(IBOUND-1) + 1
```

```
          IBNDG2(2,NBNDG2)        = ILVLG2(2,-1) + NXCELL*(IBOUND-1) +1
          IBNDG2(3,NBNDG2)        = ILVLG2(2,-1) + NXCELL*(IBOUND-2) +1
          IBNDG2(4,NBNDG2)        = 9
          KAUXG2(IBNDG2(2,NBNDG2))= IOR(KAUXG2(IBNDG2(2,NBNDG2)),KL0009)
          KAUXG2(IBNDG2(3,NBNDG2))= IOR(KAUXG2(IBNDG2(3,NBNDG2)),KL0009)
110     CONTINUE
C
C       CORRECT THE NEIGHBOUR BOUNDARY CORNER POINTER OF THE FIRST NODE
C
        NBCPG2(1,1)             = NBNDG2
C
C       INITIALIZE THE NEIGHBOUR CELL ARRAY
C
        DO 120 K = 1, 4
          DO 120 KN = 1, MNODG2
            NEIBG2(K,KN) = 0
120     CONTINUE

        DO 130 LCELL = 1, NCELG2
           KSW           = ICELG2(2,LCELL)
           KSE           = ICELG2(4,LCELL)
           KNE           = ICELG2(6,LCELL)
           KNW           = ICELG2(8,LCELL)
           NEIBG2(1,KNE) = LCELL
           NEIBG2(2,KNW) = LCELL
           NEIBG2(3,KSW) = LCELL
           NEIBG2(4,KSE) = LCELL
130     CONTINUE
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(12I5)
1100    FORMAT(4G16.7)
C
        RETURN
        END
```

G2LCAT

```
        SUBROUTINE  G2LCAT (IBOUND, XPNT, YPNT)

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'JACOMN.INC'
        DIMENSION RECT(2,5), DPENVA(5,MEQNFL)

C***********************************************************************

C       THIS SUBROUTINE DETERMINES IF THE TEST POINT (XPNT,YPNT) NEAR
```

```
C       THE BOUNDARY NODE 'IBOUND' BELONGS TO EITHER OF THE NEIGHOURING
C       CELLS IONE OR ITWO.  IF THE POINT IS NOT LOCATED IN THESE CELLS
C       THEN AN ERROR CONDITION OCCURS OTHERWISE A QUADRATIC INTERPOLATION
C       BASED ON THE CORNER NODES OF THE LOCATED CELL IS DONE AT THE
C       GIVEN POINT.

C***********************************************************************
C
C       INITIALIZATION
C
        IONE = IBNDG2(2,IBOUND)
        ITWO = IBNDG2(3,IBOUND)
C
C       CHECK CELL IONE
C
        DO 10 IP = 1, 4
           IP2 = 2*IP
           RECT(1,IP) = GEOMG2(1, ICELG2(IP2,IONE) )
           RECT(2,IP) = GEOMG2(2, ICELG2(IP2,IONE) )
10      CONTINUE
C
        CALL INSIDE (IN, RECT, 4, XPNT, YPNT)
        ICELL = IONE
C
C       NOW CHECK CELL ITWO IF NEED BE
C
        IF (IN .EQ. 0 .AND. ITWO .NE. 0) THEN
           DO 20 IP = 1, 4
              IP2 = 2*IP
              RECT(1,IP) = GEOMG2(1, ICELG2(IP2,ITWO) )
              RECT(2,IP) = GEOMG2(2, ICELG2(IP2,ITWO) )
20         CONTINUE
           CALL INSIDE (IN, RECT, 4, XPNT, YPNT)
           ICELL = ITWO
        ENDIF
C
C       ERROR CONDITION
C
        IF (IN ..EQ. 0) THEN
           INODE = IBNDG2(1,IBOUND)
           IEDGE = IBNDG2(4,IBOUND)
           ITYPE = IBNDG2(5,IBOUND)
           ZER1  = IONE
           ZER2  = ITWO
           WRITE(JPRINT,1000) IBOUND, INODE, IEDGE, ITYPE, IONE, ITWO,
     1                        XPNT  , YPNT
           KSW   = ICELG2(2,IONE)
           KSE   = ICELG2(4,IONE)
           KNE   = ICELG2(6,IONE)
           KNW   = ICELG2(8,IONE)
           XSW   = GEOMG2(1,KSW)
           XSE   = GEOMG2(1,KSE)
           XNE   = GEOMG2(1,KNE)
           XNW   = GEOMG2(1,KNW)
           YSW   = GEOMG2(2,KSW)
           YSE   = GEOMG2(2,KSE)
           YNE   = GEOMG2(2,KNE)
```

```
            YNW   = GEOMG2(2,KNW)
            ICELL = IONE
            WRITE(JPRINT,1100) IONE, KSW, XSW, YSW, KSE, XSE, YSE,
     1                                 KNE, XNE, YNE, KNW, XNW, YNW
            IF (ITWO .NE. 0) THEN
               XC  = 0.25*(XSW + XSE + XNE + XNW)
               YC  = 0.25*(YSW + YSE + YNE + YNW)
               DC1 = SQRT( (XC-XPNT)**2 + (YC-YPNT)**2 )
               KSW = ICELG2(2,ITWO)
               KSE = ICELG2(4,ITWO)
               KNE = ICELG2(6,ITWO)
               KNW = ICELG2(8,ITWO)
               XSW = GEOMG2(1,KSW)
               XSE = GEOMG2(1,KSE)
               XNE = GEOMG2(1,KNE)
               XNW = GEOMG2(1,KNW)
               YSW = GEOMG2(2,KSW)
               YSE = GEOMG2(2,KSE)
               YNE = GEOMG2(2,KNE)
               YNW = GEOMG2(2,KNW)
               XC  = 0.25*(XSW + XSE + XNE + XNW)
               YC  = 0.25*(YSW + YSE + YNE + YNW)
               DC2 = SQRT( (XC-XPNT)**2 + (YC-YPNT)**2 )
               IF (DC2 .LT. DC1) ICELL = ITWO
               WRITE(JPRINT,1200) ITWO, KSW, XSW, YSW, KSE, XSE, YSE,
     1                                  KNE, XNE, YNE, KNW, XNW, YNW
            ENDIF

            CALL WARNIN (43,'G2LCAT','IONE  ',ZER1,'ITWO  ',ZER2,JPRINT,
     1          'THE POINT IN QUESTION IS IN NEITHER OF THE TWO CELLS')
         ENDIF
C
C        SET THE POINT WHERE INTERPOLATION IS DESIRED
C
         RECT(1,5) = XPNT
         RECT(2,5) = YPNT

         DO 40 IN = 1, NEQNFL
            DO 30 IP = 1, 4
               IP2 = 2*IP
               DPENVA(IP,IN) = DPENG2(IN, ICELG2(IP2,ICELL) )
30          CONTINUE
40       CONTINUE
C
C        INTERPOLATE THE VALUES
C
         CALL INTERP (RECT, DPENVA, NEQNFL)
C
C        SET THE VALUES OF THE INTERPOLATED POINT
C
         DO 50 IN = 1, NEQNFL
             DPENJA(IN) = DPENVA(5,IN)
             BIGWJA(IN) = 0.
50       CONTINUE
C
C        ----------------
C        FORMAT STATEMENTS
```

```
C       ----------------
C
1000    FORMAT(5X,'IBOUND =',I5,   7X,'INODE =',I5,5X,'IEDGE =',I5/
     1         5X,'ITYPE  =',I5,   7X,'IONE  =',I5,5X,'ITWO  =',I5/
     2         5X,'XPNT   =',G10.3,2X,'YPNT  =',G10.3)
1100    FORMAT(5X,'IONE =',I5,3X,'SW',I17,2G14.5/
     1                      19X,'SE',I17,2G14.5 /
     2                      19X,'NE',I17,2G14.5 /
     2                      19X,'NW',I17,2G14.5 )
1200    FORMAT(5X,'ITWO =',I5,3X,'SW',I17,2G14.5/
     1                      19X,'SE',I17,2G14.5 /
     2                      19X,'NE',I17,2G14.5 /
     2                      19X,'NW',I17,2G14.5 )
C
        RETURN
        END
```

G2NODE

```
        SUBROUTINE G2NODE

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'

C***********************************************************************

C       THIS SUBROUTINE COLLECTS ALL THE NODES POINTERS (INTERRIOR AND
C       BOUNDARY) MARKED FOR DELETE BY SUBROUTINE G2CLP0 OR G2CLP1.
C       NOTE THAT THE CELL POINTERS REMAIN THE SAME WHEREAS THE NODE
C       POINTERS ARE REALLIGNED.

C***********************************************************************
C
C       COUNT THE NUMBER OF NODES TO BE DELETED AND INITIALIZE THE
C       LIST MRKDA2 OF NODES NOT TO BE DELETED
C
        NDEL = 0
        DO 10 NOLD = 1, NNODG2
          MRKDA2(NOLD) = 0
          IF (DPENG2(1,NOLD) .EQ. -99.) NDEL = NDEL + 1
10      CONTINUE

        IF (NDEL .EQ. 0) RETURN

C       DELETE ALL NODES MARKED FOR DELETE AND MOVE NODE INFORMATION

        NNEW = 0
        DO 50 NOLD = 1, NNODG2
          IF (DPENG2(1,NOLD) .NE. -99.) THEN
             NNEW        = NNEW + 1
```

```
            MRKDA2(NOLD) = NNEW
           IF (NOLD .NE. NNEW) THEN
C             ADJUST THE GEOMETRY ARRAYS AT THE MOVED NODES
              DO 20 J = 1, 2
                GEOMG2(J,NNEW) = GEOMG2(J,NOLD)
20            CONTINUE
C             ADJUST THE DEPENDENT VARIABLES
              DO 30 J = 1, NEQNFL
                 DPENG2(J,NNEW) = DPENG2(J,NOLD)
30            CONTINUE
C             ADJUST THE PRESSURE & TEMPERATURE
              PRESG2(NNEW) = PRESG2(NOLD)
              TEMPG2(NNEW) = TEMPG2(NOLD)
C             ADJUST THE NEIGHBOUR-NODE-ARRAYS
              DO 40 J = 1, 4
                 NEIBG2(J,NNEW) = NEIBG2(J,NOLD)
40            CONTINUE
            ENDIF
          ENDIF
50      CONTINUE

C       RESET NUMBER OF NODES

        NNODG2 = NNEW

C       DELETE ALL BOUNDARY CONDITION POINTERS MARKED FOR DELETE
C
        NNEW = 0
        DO 70 NOLD = 1, NBNDG2
          MRKCA2(NOLD) = 0
          IF (IBNDG2(1,NOLD) .NE. -9) THEN
             NNEW = NNEW + 1
             MRKCA2(NOLD) = NNEW
C            MOVE POINTER INFORMATION
             IF (NOLD .NE. NNEW) THEN
                DO 60 J = 1, 5
                   IBNDG2(J,NNEW) = IBNDG2(J,NOLD)
60              CONTINUE
             ENDIF
          ENDIF
70      CONTINUE
C
C       RESET NUMBER OF BOUNDARY CONDITION POINTERS
C
        NBNDG2 = NNEW

C       STEP THROUGH ALL CELL POINTERS, WHICH POINT TOWARDS NODES,
C       REALIGNING TO NEW NODE NUMBERS.  THE NODE NUMBERS CORRESPONDING
C       TO COARSE CELLS ARE NOT CHANGED

        DO 90 ICELL = ILVLG2(1,0), NCELG2
C         STEP THROUGH EACH CELL POINTER
          DO 80 IPNT = 1, 9
            IF (ICELG2(IPNT,ICELL) .NE. 0) THEN
                ICELG2(IPNT,ICELL) = MRKDA2(ICELG2(IPNT,ICELL))
            ENDIF
80        CONTINUE
```

```
90      CONTINUE
C
C       STEP THROUGH ALL BOUNDARY CONDITION POINTERS, REALIGNING TO
C       NEW NODE NUMBERS
C
        DO 100 IBND = 1, NBNDG2
          IF (IBNDG2(1,IBND).NE.-9)
     1        IBNDG2(1,IBND)=MRKDA2(IBNDG2(1,IBND))
100     CONTINUE

        DO 120 IEDGE = 1, 4
          DO 110 IBND = 1, 2
            NBCPG2(IEDGE,IBND) = MRKCA2(NBCPG2(IEDGE,IBND))
110       CONTINUE
120     CONTINUE

        RETURN
        END
```

G2PRNT

```
        SUBROUTINE G2PRNT (IOPT)

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'HEXCOD.INC'
        INCLUDE 'IOCOMN.INC'

C*****************************************************************************

C       THIS SUBROUTINE PRINTS ALL ARRAY VARIABLES.
C       IOPT SELECTS WHICH TABLES ARE TO BE PRINTED:
C
C         B.CODE      IOPT   NODE  CELL   BNDY   AUX
C          0000         0     X     X     X      X
C          0001         1           X     X      X
C          0010         2     X           X      X
C          0011         3                 X      X
C          0100         4     X     X             X
C          0101         5           X             X
C          0110         6     X                   X
C          0111         7                         X
C          1000         8     X     X     X
C          1001         9           X     X
C          1010        10     X           X
C          1011        11                 X
C          1100        12     X     X
C          1101        13           X
C          1110        14     X
C          1111        15     ALL COMPACT FORM
C
C*****************************************************************************
```

```
      IF (IOPT .EQ. 15) GOTO 160
C
C     --------------
C     NODE VARIABLES
C     --------------
C     WRITE OUT NODE ARRAYS

      IF (IAND(IOPT,KL0001) .EQ. 0) THEN

C       CONDITION IS MET IF IOPT = 0,2,4,6,8,10,12,14

        CALL HEADER(JPRINT,'NODE VARIABLES-- GEOMETRY AND NEIGHBOUR',
     1              MTITLE)

        WRITE(JPRINT,10)
10      FORMAT(7X,'NODE ',9X,'GEOM1',10X,'GEOM2',11X,
     1           'NBSW ',2X,'NBSE ', 2X,'NBNE ', 2X,'NBNW')

        DO 30 INODE = 1, NNODG2
          WRITE(JPRINT,20) INODE, (GEOMG2(K,INODE),K=1,2),
     1                            (NEIBG2(K,INODE),K=1,4)
20        FORMAT(1X,(I9,6X),2G15.8,2X,4I7)
30      CONTINUE

        CALL HEADER(JPRINT,'DEPENDENT NODE VARIABLES',MTITLE)

        WRITE(JPRINT,40)

40      FORMAT(7X,'NODE',10X,'DEPENDENT VARIABLES')

        NT = MAX (8, NEQNFL)

        DO 60 INODE = 1, NNODG2
          WRITE(JPRINT,50) INODE, (DPENG2(K,INODE),K=1,NEQNFL)
50        FORMAT(1X,(I9,6X),8G14.6)
60      CONTINUE

      ENDIF
C
C     --------------
C     CELL VARIABLES
C     --------------

C     WRITE OUT CELL ARRAYS

      IF (IAND(IOPT,KL0002) .EQ. 0) THEN

C       CONDITION IS MET IF IOPT = 0,1,4,5,8,9,12,13

        CALL HEADER(JPRINT,'CELL VARIABLES',MTITLE)

        WRITE(JPRINT, 70)
70      FORMAT(5X,'CELL ',5X,'CENT ',5X,'SWEST',5X,'SOUTH',5X,'SEAST',
     1         5X,'EAST ',5X,'NEAST',5X,'NORTH',5X,'NWEST',5X,'WEST ',
     2         5X,'SUPER',7X,'AUXIL')
```

```
          DO 90 ICELL = 1, NCELG2
            WRITE(JPRINT,80) ICELL,
     1              (ICELG2(K,ICELL),K=1,10),KAUXG2(ICELL)
80          FORMAT(1X,11(I7,3X),Z10)
90        CONTINUE

        ENDIF

C       --------------------
C       BOUNDARY INFORMATION
C       --------------------
C
C       WRITE OUT BOUNDARY CONDITION ARRAYS
C
        IF (IAND(IOPT,KL0004) .EQ. 0) THEN

C         CONDITION IS MET IF IOPT = 0,1,2,3,8,9,10,11

C         BOUNDARY CONDITION ARRAYS

          CALL HEADER(JPRINT,'BOUNDARY CONDITION INFORMATION',MTITLE)

          WRITE(JPRINT,100)
100       FORMAT(5X,'BOUND',5X,'NODE ',5X,'CELL1',5X,'CELL2',
     1           5X,'EDGE ',5X,'TYPE')

          DO 120 IBOUND = 1, NBNDG2
            WRITE(JPRINT,110) IBOUND, (IBNDG2(K,IBOUND),K=1,5)
110         FORMAT(1X,6(I7,3X))
120       CONTINUE

        ENDIF

C       --------------------
C       AUXILIARY INFORMATION
C       --------------------

        IF (IAND(IOPT,KL0008) .EQ. 0) THEN

C         CONDITION IS MET IF IOPT = 0,1,2,3,4,5,6,7

C         MULTIPLE-GRID-LEVEL ARRAY

          WRITE(JPRINT,130)
130       FORMAT(//' MULTIPLE-GRID-LEVEL INFORMATION:'//
     1             5X,'LEVEL',4X,'START',6X,'END',5X,'# CELLS')

          DO 150 IMGL = -MLVLG2, MLVLG2
            WRITE(JPRINT,140) IMGL,(ILVLG2(K,IMGL),K=1,3)
140         FORMAT(1X,4(I7,3X))
150       CONTINUE

        ENDIF

        RETURN

160     CONTINUE
```

```
      WRITE(JPRINT,*) ' NNODG2 = ',NNODG2
      WRITE(JPRINT,170)
170   FORMAT(4X,'CELL ',2X,'CENT ',2X,'SWEST',2X,'SOUTH',2X,'SEAST',
     1       2X,'EAST ',2X,'NEAST',2X,'NORTH',2X,'NWEST',2X,'WEST ',
     2       2X,'SUPER',5X,'AUXIL',2X,'NBSW ',2X,'NBSE ',2X,'NBNE ',
     3       2X,'NBNW ',2X,'GEOM1')

      DO 190 ICELL = 1, NCELG2
        WRITE(JPRINT,180) ICELL,(ICELG2(K,ICELL),K=1,10),
     1    KAUXG2(ICELL), (NEIBG2(K,ICELL),K=1,4),GEOMG2(1,ICELL)
180     FORMAT(11(2X,I5),Z10,4(2X,I5),F10.3)
190   CONTINUE

      END
```

G2RES0

```
      SUBROUTINE G2RES0

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'PRCOMN.INC'
      CHARACTER RECORD*132

C*********************************************************************

C     THIS SUBROUTINE WRITES THE FINAL RESULTS FOR THE UNSTEADY
C     FLOW PROBLEMS

C*********************************************************************

      WRITE(JOUTAL,1000)
      WRITE(JOUTAL,1100) NITRE2, NNODG2, NCELG2, NBNDG2, NCELA2

      NT = MIN (8, NEQNFL)
      LT = 0
      IF (NEQNFL .GT. NT) LT = NEQNFL - NT

      RECORD        = ' '
      RECORD( 2: 5) = 'NODE'
      RECORD( 9:18) = 'X-DISTANCE'
      RECORD(20:29) = 'Y-DISTANCE'
      IBG           = 32

      DO 10 N = 1, NT
        IED             = IBG + 4
        RECORD(IBG:IED) = 'DEPEN'
        IED             = IED + 1
```

```
          IBG             = IED + 8
          WRITE(RECORD(IED:IED),1200) N
10      CONTINUE

        WRITE(JOUTAL, 1300) RECORD

        DO 20 I = 1, NNODG2
          WRITE(JOUTAL,1400) I,(GEOMG2(J,I),J=1,2),(DPENG2(J,I),J=1,NT)
20      CONTINUE

        IF (LT .NE. 0) THEN
          IBG           = 33
          RECORD(IBG:132) = ' '
          DO 30 N = 1, NT
            IED             = IBG + 4
            RECORD(IBG:IED) = 'DEPEN'
            IED             = IED + 1
            IBG             = IED + 7
            WRITE(RECORD(IED:IED),1500) N
30        CONTINUE
          WRITE(JOUTAL,1600) RECORD
          DO 40 I = 1, NNODG2
            WRITE(JOUTAL,1400) I, (GEOMG2(J,I),J=1,2),
     1                            (DPENG2(J,I),J=NT+1,NEQNFL)
40        CONTINUE
        ENDIF

        WRITE(JOUTAL,1700)
        WRITE(JOUTAL,1800)

        OPEN (UNIT=JDUMY1, STATUS='SCRATCH')

        DO 50 I = 1, NNODG2
          CALL   E2PRMT (I,3)
          STAGHS = (BEPSPR + PRESPR)/RHORPR
          WRITE(JOUTAL,1400) I, (GEOMG2(J,I),J=1,2), UCOMPR, VCOMPR,
     1           TEMPPR, PRESPR, AMCHPR, GAMAPR, BEPSPR, STAGHS
          WRITE(JDUMY1,1400) I, (GEOMG2(J,I),J=1,2),
     1                          (YSPEPR(IS),IS=1,NSPECH)
50      CONTINUE

        REWIND (JDUMY1)
        RECORD        = ' '
        RECORD( 2: 5) = 'NODE'
        RECORD( 9:18) = 'X-DISTANCE'
        RECORD(20:29) = 'Y-DISTANCE'
        IBG           = 33
        NT            = MIN (12, NSPECH)

        DO 60 N = 1, NT
          IED             = IBG + 4
          RECORD(IBG:IED) = 'FRACT'
          IED             = IED + 1
          IBG             = IED + 5
          WRITE(RECORD(IED:IED),1200) N
60      CONTINUE
```

```
      WRITE(JOUTAL,1900) RECORD
        -
      DO 70 I = 1, NNODG2
        READ (JDUMY1,1400) K,(GEOMG2(J,I),J =1,2     ),
     1                       (YSPEPR(IS ),IS=1,NSPECH)
        WRITE(JOUTAL,1400) K,(GEOMG2(J,I),J =1,2     ),
     1                       (YSPEPR(IS ),IS=1,NSPECH)
70    CONTINUE
C
C     -----------------
C     FORMAT STATEMENTS
C     -----------------
C
1000  FORMAT('1'//)
1100  FORMAT(5X,'TOTAL NUMBER OF ITERATIONS     = ',I5,10X,
     1       5X,'TOTAL NUMBER OF NODES          = ',I5 /
     2       5X,'TOTAL NUMBER OF CELLS          = ',I5 ,10X,
     3       5X,'TOTAL NUMBER OF BOUNDARY NODES = ',I5 /
     4       5X,'TOTAL NUMBER OF CEWIC CELLS    = ',I5 /)
1200  FORMAT(I1)
1300  FORMAT(/'1',10X,'GEOMETRY AND DEPENDENT VARIABLES'//A130)
1400  FORMAT (I6,1X,2F10.5,8G13.5)
1500  FORMAT(I2)
1600  FORMAT(/'1','CONT'//,A132)
1700  FORMAT(/'1',10X,'PRIMITIVE VARIABLES'/)
1800  FORMAT(2X, 'NODE', 1X, 'X-DISTANCE', 2X,'Y-DISTANCE', 2X,
     1    'UCOMPON', 5X, 'VCOMPON', 6X, 'TEMPER', 6X, 'PRESSURE', 6X,
     2    'MACH NO.', 4X, 'GAMMA', 7X, 'T. ENERGY', 3X, 'STAG ENTH'/ )
1900  FORMAT(/'1',10X,'CONCENTRATION VARIABLES'//A130)

      RETURN
      END
```

G2SMOT

```
      SUBROUTINE G2SMOT

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'G2COMN.INC'
      DIMENSION DPLEFT(MEQNFL), DPRITE(MEQNFL)
      DIMENSION DPBOT (MEQNFL), DPTOP (MEQNFL)
C     DATA SMALLP /1.E-10/, SMALLN /-1.E-10/
      DATA SMALLP /1.E-3/, SMALLN /-1.E-3/

C*********************************************************************

C     THIS SUBROUTINE CORRECTS THE CONSERVATIVE VARIABLES AT A GIVEN
C     NODE 'INODE', IF THERE ARE OSCILLATIONS AT A NODE.  THE OSCILL-
C     ATIONS ARE DEFINED TO BE THE ONES WHICH CAUSE DISCONTINOUS
C     FIRST DIFFERENCES AT A NODE.  IT IS HOPED THAT SUCH A SITUATION
C     ONLY OCCURS AT A FEW NODES.
```

```
C*****************************************************************
      DO 100 INODE = 1, NNODG2
         NB1 = NEIBG2(1,INODE)
         NB2 = NEIBG2(2,INODE)
         NB3 = NEIBG2(3,INODE)
         NB4 = NEIBG2(4,INODE)

C        THE CORRECTION IS NOT APPLIED AT THE CORNER BOUNDARY CELLS

c        IF (NB1 .EQ. 0 .AND. NB4 .EQ. 0) GOTO 100
c        IF (NB2 .EQ. 0 .AND. NB3 .EQ. 0) GOTO 100
         IF (NB1 .EQ. 0 .or. NB2 .EQ. 0 .or.
     1       NB3 .EQ. 0 .or. NB4 .EQ. 0     ) GOTO 100

C        SETUP THE LEFT NEIGHBOUR CELL AND ITS NODE POINTER

         IF (NB1 .NE. 0) THEN
           NBLEFT = NB1
           IPLEFT = 8
         ELSE
           NBLEFT = NB4
           IPLEFT = 2
         ENDIF
C
C        FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
         IF (NB1 .EQ. NB4) THEN
           IPLEFT = 9
           IF (ICELG2(IPLEFT,NBLEFT) .EQ. 0) THEN
             INLFT1 = ICELG2(2,NBLEFT)
             INLFT2 = ICELG2(8,NBLEFT)
             XLEFT  = 0.5*(GEOMG2(1,INLFT1)+GEOMG2(1,INLFT2))
             YLEFT  = 0.5*(GEOMG2(2,INLFT1)+GEOMG2(2,INLFT2))
             DO 10 IQ = 1, NEQNFL
               DPLEFT(IQ) = 0.5*(DPENG2(IQ,INLFT1)+DPENG2(IQ,INLFT2))
10           CONTINUE
             GOTO 30
           ENDIF
         ENDIF

C        COMPUTE THE LEFT NODE, DISTANCES AND DP VARIABLES

         INLEFT = ICELG2(IPLEFT,NBLEFT)

         XLEFT  = GEOMG2(1,INLEFT)
         YLEFT  = GEOMG2(2,INLEFT)
         DO 20 IQ = 1, NEQNFL
            DPLEFT(IQ) = DPENG2(IQ,INLEFT)
20       CONTINUE

30       CONTINUE

C        SETUP THE RIGHT NEIGHBOUR CELL AND ITS NODE POINTER

         IF (NB2 .NE. 0) THEN
```

```
            NBRITE = NB2
            IPRITE = 6
          ELSE
            NBRITE = NB3
            IPRITE = 4
          ENDIF
C
C         FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
          IF (NB2 .EQ. NB3) THEN
            IPRITE = 5
            IF (ICELG2(IPRITE,NBRITE) .EQ. 0) THEN
              INRIT1 = ICELG2(4,NBRITE)
              INRIT2 = ICELG2(6,NBRITE)
              XRITE  = 0.5*(GEOMG2(1,INRIT1)+GEOMG2(1,INRIT2))
              YRITE  = 0.5*(GEOMG2(2,INRIT1)+GEOMG2(2,INRIT2))
              DO 40 IQ = 1, NEQNFL
                DPRITE(IQ) = 0.5*(DPENG2(IQ,INRIT1)+DPENG2(IQ,INRIT2))
40            CONTINUE
              GOTO 60
            ENDIF
          ENDIF

C         COMPUTE THE RIGHT NODE, DISTANCES AND DP VARIABLES

          INRITE = ICELG2(IPRITE,NBRITE)
          XRITE  = GEOMG2(1,INRITE)
          YRITE  = GEOMG2(2,INRITE)

          DO 50 IQ = 1, NEQNFL
             DPRITE(IQ) = DPENG2(IQ,INRITE)
50        CONTINUE

60        CONTINUE

C         NOW CHECK FOR DENSITY DIFFERENCES ACROSS THE NODE

          DDLEFT = DPENG2(1,INODE) - DPLEFT(1)
          DDRITE = DPENG2(1,INODE) - DPRITE(1)
          IF (DDLEFT .GT. SMALLP .AND. DDRITE .GT. SMALLP) GOTO 70
          IF (DDLEFT .LT. SMALLN .AND. DDRITE .LT. SMALLN) GOTO 70
          GO TO 100

70        XNODE  = GEOMG2(1,INODE)
          YNODE  = GEOMG2(2,INODE)
          SNODE2 = (XNODE-XLEFT)**2 + (YNODE-YLEFT)**2
          SRITE2 = (XRITE-XLEFT)**2 + (YRITE-YLEFT)**2
          RATIO  = SQRT(SNODE2/SRITE2)
C
C         DO THE INTERPOLATION
C
          DO 80 IQ = 1, NEQNFL
             DPHERE = DPLEFT(IQ) + (DPRITE(IQ) -DPLEFT(IQ))*RATIO
             DPENG2(IQ,INODE) = 0.5*(DPHERE + DPENG2(IQ,INODE))
80        CONTINUE
C
C         NOW RECOMPUTE THE PRESSURE, TEMPERATURE ETC.
```

```
C
         CALL E2PRMT(INODE,1)
100      CONTINUE

C        NOW REPEAT THE WHOLE PROCESS FOR Y-AXIS

         DO 200 INODE = 1, NNODG2
            NB1 = NEIBG2(1,INODE)
            NB2 = NEIBG2(2,INODE)
            NB3 = NEIBG2(3,INODE)
            NB4 = NEIBG2(4,INODE)

C           THE CORRECTION IS NOT APPLIED AT THE CORNER BOUNDARY CELLS

c           IF (NB1 .EQ. 0 .AND. NB2 .EQ. 0) GOTO 200
c           IF (NB3 .EQ. 0 .AND. NB4 .EQ. 0) GOTO 200
            IF (NB1 .EQ. 0 .or. NB2 .EQ. 0 .or.
     1          NB3 .EQ. 0 .or. NB4 .EQ. 0     ) GOTO 200

C           SETUP THE BOTTOM NEIGHBOUR CELL AND ITS NODE POINTER

            IF (NB1 .NE. 0) THEN
              NBBOT = NB1
              IPBOT = 4
            ELSE
              NBBOT = NB2
              IPBOT = 2
            ENDIF
C
C           FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
            IF (NB1 .EQ. NB2) THEN
              IPBOT = 3
              IF (ICELG2(IPBOT,NBBOT) .EQ. 0) THEN
                INBOT1 = ICELG2(2,NBBOT)
                INBOT2 = ICELG2(4,NBBOT)
                XBOT  = 0.5*(GEOMG2(1,INBOT1)+GEOMG2(1,INBOT2))
                YBOT  = 0.5*(GEOMG2(2,INBOT1)+GEOMG2(2,INBOT2))
                DO 110 IQ = 1, NEQNFL
                  DPBOT(IQ) = 0.5*(DPENG2(IQ,INBOT1)+DPENG2(IQ,INBOT2))
110             CONTINUE
                GOTO 130
              ENDIF
            ENDIF

C           COMPUTE THE BOTTOM NODE, DISTANCES AND DP VARIABLES

            INBOT = ICELG2(IPBOT,NBBOT)

            XBOT  = GEOMG2(1,INBOT)
            YBOT  = GEOMG2(2,INBOT)
            DO 120 IQ = 1, NEQNFL
               DPBOT(IQ) = DPENG2(IQ,INBOT)
120         CONTINUE

130         CONTINUE
```

```
C         SETUP THE TOP NEIGHBOUR CELL AND ITS NODE POINTER
          IF (NB3 .NE. 0) THEN
            NBTOP = NB3
            IPTOP = 8
          ELSE
            NBTOP = NB4
            IPTOP = 6
          ENDIF
C
C         FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
          IF (NB3 .EQ. NB4) THEN
            IPTOP = 7
            IF (ICELG2(IPTOP,NBTOP) .EQ. 0) THEN
              INTOP1 = ICELG2(6,NBTOP)
              INTOP2 = ICELG2(8,NBTOP)
              XTOP  = 0.5*(GEOMG2(1,INTOP1)+GEOMG2(1,INTOP2))
              YTOP  = 0.5*(GEOMG2(2,INTOP1)+GEOMG2(2,INTOP2))
              DO 140 IQ = 1, NEQNFL
                DPTOP(IQ) = 0.5*(DPENG2(IQ,INTOP1)+DPENG2(IQ,INTOP2))
140           CONTINUE
              GOTO 160
            ENDIF
          ENDIF

C         COMPUTE THE TOP NODE, DISTANCES AND DP VARIABLES

          INTOP = ICELG2(IPTOP,NBTOP)
          XTOP  = GEOMG2(1,INTOP)
          YTOP  = GEOMG2(2,INTOP)

          DO 150 IQ = 1, NEQNFL
             DPTOP(IQ) = DPENG2(IQ,INTOP)
150       CONTINUE

160       CONTINUE

C         NOW CHECK FOR DENSITY DIFFERENCES ACROSS THE NODE

          DDBOT = DPENG2(1,INODE) - DPBOT(1)
          DDTOP = DPENG2(1,INODE) - DPTOP(1)
          IF (DDBOT .GT. SMALLP .AND. DDTOP .GT. SMALLP) GOTO 170
          IF (DDBOT .LT. SMALLN .AND. DDTOP .LT. SMALLN) GOTO 170
          GO TO 200

170       XNODE  = GEOMG2(1,INODE)
          YNODE  = GEOMG2(2,INODE)
          SNODE2 = (XNODE-XBOT)**2 + (YNODE-YBOT)**2
          STOP2 = (XTOP-XBOT)**2 + (YTOP-YBOT)**2
          RATIO  = SQRT(SNODE2/STOP2)
C
C         DO THE INTERPOLATION
C
          DO 180 IQ = 1, NEQNFL
             DPHERE = DPBOT(IQ) + (DPTOP(IQ) -DPBOT(IQ))*RATIO
             DPENG2(IQ,INODE) = 0.5*(DPHERE + DPENG2(IQ,INODE))
```

```
180         CONTINUE
C
C           NOW RECOMPUTE THE PRESSURE, TEMPERATURE ETC.
C
            CALL E2PRMT(INODE,1)
200      CONTINUE

         RETURN
         END
```

G3SMOT

```
         SUBROUTINE G3SMOT

         INCLUDE 'PRECIS.INC'
         INCLUDE 'PARMV2.INC'
         INCLUDE 'G2COMN.INC'
         INCLUDE 'TICOMN.INC'
         DIMENSION DPLEFT(MEQNFL), DPRITE(MEQNFL)
         DIMENSION DPBOT (MEQNFL), DPTOP (MEQNFL)
         DATA SMALLP /1.E-4/, SMALLN /-1.E-4/
C        DATA SMALLP /1.E-3/, SMALLN /-1.E-3/

C*********************************************************************

C        THIS SUBROUTINE CORRECTS THE CONSERVATIVE VARIABLES AT A GIVEN
C        NODE 'INODE', IF THERE ARE OSCILLATIONS AT A NODE.  THE OSCILL-
C        ATIONS ARE DEFINED TO BE THE ONES WHICH CAUSE DISCONTINOUS
C        FIRST DIFFERENCES AT A NODE.  IT IS HOPED THAT SUCH A SITUATION
C        ONLY OCCURS AT A FEW NODES.

C*********************************************************************

         IF (KADPTI .LE. NEQBAS .OR. KADPTI .GT. NEQNFL) RETURN

         DO 100 INODE = 1, NNODG2
            NB1 = NEIBG2(1,INODE)
            NB2 = NEIBG2(2,INODE)
            NB3 = NEIBG2(3,INODE)
            NB4 = NEIBG2(4,INODE)

C           THE CORRECTION IS NOT APPLIED AT THE CORNER BOUNDARY CELLS

            IF (NB1 .EQ. 0 .AND. NB4 .EQ. 0) GOTO 100
            IF (NB2 .EQ. 0 .AND. NB3 .EQ. 0) GOTO 100
c           IF (NB1 .EQ. 0 .or. NB2 .EQ. 0 .or.
c    1              NB3 .EQ. 0 .or. NB4 .EQ. 0     ) GOTO 100

C           SETUP THE LEFT NEIGHBOUR CELL AND ITS NODE POINTER

            IF (NB1 .NE. 0) THEN
               NBLEFT = NB1
```

```
            IPLEFT = 8
          ELSE
            NBLEFT = NB4
            IPLEFT = 2
          ENDIF
C
C         FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
          IF (NB1 .EQ. NB4) THEN
            IPLEFT = 9
            IF (ICELG2(IPLEFT,NBLEFT) .EQ. 0) THEN
               INLFT1 = ICELG2(2,NBLEFT)
               INLFT2 = ICELG2(8,NBLEFT)
               XLEFT  = 0.5*(GEOMG2(1,INLFT1)+GEOMG2(1,INLFT2))
               YLEFT  = 0.5*(GEOMG2(2,INLFT1)+GEOMG2(2,INLFT2))
               DO 10 IQ = NEQBAS+1, NEQNFL
                  DPLEFT(IQ) = 0.5*(DPENG2(IQ,INLFT1)/DPENG2(1,INLFT1)
     1                             +DPENG2(IQ,INLFT2)/DPENG2(1,INLFT2))
10             CONTINUE
               GOTO 30
            ENDIF
          ENDIF

C         COMPUTE THE LEFT NODE, DISTANCES AND DP VARIABLES

          INLEFT = ICELG2(IPLEFT,NBLEFT)

          XLEFT  = GEOMG2(1,INLEFT)
          YLEFT  = GEOMG2(2,INLEFT)
          DO 20 IQ = NEQBAS+1, NEQNFL
              DPLEFT(IQ) = DPENG2(IQ,INLEFT)/DPENG2(1,INLEFT)
20        CONTINUE

30        CONTINUE

C         SETUP THE RIGHT NEIGHBOUR CELL AND ITS NODE POINTER

          IF (NB2 .NE. 0) THEN
             NBRITE = NB2
             IPRITE = 6
          ELSE
             NBRITE = NB3
             IPRITE = 4
          ENDIF
C
C         FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
          IF (NB2 .EQ. NB3) THEN
             IPRITE = 5
             IF (ICELG2(IPRITE,NBRITE) .EQ. 0) THEN
                INRIT1 = ICELG2(4,NBRITE)
                INRIT2 = ICELG2(6,NBRITE)
                XRITE  = 0.5*(GEOMG2(1,INRIT1)+GEOMG2(1,INRIT2))
                YRITE  = 0.5*(GEOMG2(2,INRIT1)+GEOMG2(2,INRIT2))
                DO 40 IQ = NEQBAS+1, NEQNFL
                  DPLEFT(IQ) = 0.5*(DPENG2(IQ,INRIT1)/DPENG2(1,INRIT1)
     1                             +DPENG2(IQ,INRIT2)/DPENG2(1,INRIT2))
```

```
40              CONTINUE
              -   GOTO 60
               ENDIF
             ENDIF

C            COMPUTE THE RIGHT NODE, DISTANCES AND DP VARIABLES

             INRITE = ICELG2(IPRITE,NBRITE)
             XRITE  = GEOMG2(1,INRITE)
             YRITE  = GEOMG2(2,INRITE)

             DO 50 IQ = NEQBAS+1, NEQNFL
                DPRITE(IQ) = DPENG2(IQ,INRITE)/DPENG2(1,INRITE)
50           CONTINUE

60           CONTINUE

C            NOW CHECK FOR DENSITY DIFFERENCES ACROSS THE NODE

             DDLEFT = DPENG2(KADPTI,INODE)/DPENG2(1,INODE) -
     1                DPLEFT(KADPTI)

             DDRITE = DPENG2(KADPTI,INODE)/DPENG2(1,INODE) -
     1                DPRITE(KADPTI)

             IF (DDLEFT .GT. SMALLP .AND. DDRITE .GT. SMALLP) GOTO 70
             IF (DDLEFT .LT. SMALLN .AND. DDRITE .LT. SMALLN) GOTO 70
             GO TO 100

70           XNODE  = GEOMG2(1,INODE)
             YNODE  = GEOMG2(2,INODE)
             SNODE2 = (XNODE-XLEFT)**2 + (YNODE-YLEFT)**2
             SRITE2 = (XRITE-XLEFT)**2 + (YRITE-YLEFT)**2
             RATIO  = SQRT(SNODE2/SRITE2)
C
C            DO THE INTERPOLATION
C
             DO 80 IQ = NEQBAS+1, NEQNFL
                DPINTR = DPLEFT(IQ) + (DPRITE(IQ) -DPLEFT(IQ))*RATIO
                DPHERE = DPENG2(IQ,INODE)/DPENG2(1,INODE)
                DPENG2(IQ,INODE) = 0.5*(DPHERE+DPINTR)*DPENG2(1,INODE)
80           CONTINUE
C
C            NOW RECOMPUTE THE PRESSURE, TEMPERATURE ETC.
C
             CALL E2PRMT(INODE,1)
100       CONTINUE

C         NOW REPEAT THE WHOLE PROCESS FOR Y-AXIS

          DO 200 INODE = 1, NNODG2
             NB1 = NEIBG2(1,INODE)
             NB2 = NEIBG2(2,INODE)
             NB3 = NEIBG2(3,INODE)
             NB4 = NEIBG2(4,INODE)

C            THE CORRECTION IS NOT APPLIED AT THE CORNER BOUNDARY CELLS
```

```
c          IF (NB1 .EQ. 0 .AND. NB2 .EQ. 0) GOTO 200
c          IF (NB3 .EQ. 0 .AND. NB4 .EQ. 0) GOTO 200
           IF (NB1 .EQ. 0 .or. NB2 .EQ. 0 .or.
     1         NB3 .EQ. 0 .or. NB4 .EQ. 0     ) GOTO 200

C          SETUP THE BOTTOM NEIGHBOUR CELL AND ITS NODE POINTER

           IF (NB1 .NE. 0) THEN
              NBBOT = NB1
              IPBOT = 4
           ELSE
              NBBOT = NB2
              IPBOT = 2
           ENDIF
C
C          FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
           IF (NB1 .EQ. NB2) THEN
             IPBOT = 3
             IF (ICELG2(IPBOT,NBBOT) .EQ. 0) THEN
                INBOT1 = ICELG2(2,NBBOT)
                INBOT2 = ICELG2(4,NBBOT)
                XBOT  = 0.5*(GEOMG2(1,INBOT1)+GEOMG2(1,INBOT2))
                YBOT  = 0.5*(GEOMG2(2,INBOT1)+GEOMG2(2,INBOT2))
                DO 110 IQ = NEQBAS+1, NEQNFL
                   DPBOT(IQ) = 0.5*(DPENG2(IQ,INBOT1)/DPENG2(1,INBOT1)
     1                             +DPENG2(IQ,INBOT2)/DPENG2(1,INBOT2))
110             CONTINUE
                GOTO 130
             ENDIF
           ENDIF

C          COMPUTE THE BOTTOM NODE, DISTANCES AND DP VARIABLES

           INBOT = ICELG2(IPBOT,NBBOT)

           XBOT  = GEOMG2(1,INBOT)
           YBOT  = GEOMG2(2,INBOT)
           DO 120 IQ = NEQBAS+1, NEQNFL
              DPBOT(IQ) = DPENG2(IQ,INBOT)/DPENG2(1,INBOT)
120        CONTINUE

130        CONTINUE

C          SETUP THE TOP NEIGHBOUR CELL AND ITS NODE POINTER

           IF (NB3 .NE. 0) THEN
              NBTOP = NB3
              IPTOP = 8
           ELSE
              NBTOP = NB4
              IPTOP = 6
           ENDIF
C
C          FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
```

```
            IF (NB3 .EQ. NB4) THEN
              IPTOP = 7
              IF (ICELG2(IPTOP,NBTOP) .EQ. 0) THEN
                 INTOP1 = ICELG2(6,NBTOP)
                 INTOP2 = ICELG2(8,NBTOP)
                 XTOP  = 0.5*(GEOMG2(1,INTOP1)+GEOMG2(1,INTOP2))
                 YTOP  = 0.5*(GEOMG2(2,INTOP1)+GEOMG2(2,INTOP2))
                 DO 140 IQ = NEQBAS+1, NEQNFL
                    DPTOP(IQ) = 0.5*(DPENG2(IQ,INTOP1)/DPENG2(1,INTOP1)
     1                           +DPENG2(IQ,INTOP2)/DPENG2(1,INTOP2))
140              CONTINUE
                 GOTO 160
              ENDIF
            ENDIF

C           COMPUTE THE TOP NODE, DISTANCES AND DP VARIABLES

            INTOP = ICELG2(IPTOP,NBTOP)
            XTOP  = GEOMG2(1,INTOP)
            YTOP  = GEOMG2(2,INTOP)

            DO 150 IQ = NEQBAS+1, NEQNFL
               DPTOP(IQ) = DPENG2(IQ,INTOP)/DPENG2(1,INTOP)
150         CONTINUE

160         CONTINUE

C           NOW CHECK FOR DENSITY DIFFERENCES ACROSS THE NODE

            DDBOT = DPENG2(KADPTI,INODE)/DPENG2(1,INODE) -
     1              DPBOT(KADPTI)
            DDTOP = DPENG2(KADPTI,INODE)/DPENG2(1,INODE) -
     1              DPTOP(KADPTI)
            IF (DDBOT .GT. SMALLP .AND. DDTOP .GT. SMALLP) GOTO 170
            IF (DDBOT .LT. SMALLN .AND. DDTOP .LT. SMALLN) GOTO 170
            GO TO 200

170         XNODE  = GEOMG2(1,INODE)
            YNODE  = GEOMG2(2,INODE)
            SNODE2 = (XNODE-XBOT)**2 + (YNODE-YBOT)**2
            STOP2 = (XTOP-XBOT)**2 + (YTOP-YBOT)**2
            RATIO  = SQRT(SNODE2/STOP2)
C
C           DO THE INTERPOLATION
C
            DO 180 IQ = NEQBAS+1, NEQNFL
               DPINTR = DPBOT(IQ) + (DPTOP(IQ) -DPBOT(IQ))*RATIO
               DPHERE = DPENG2(IQ,INODE)/DPENG2(1,INODE)
               DPENG2(IQ,INODE) = 0.5*(DPHERE+DPINTR)*DPENG2(1,INODE)
180         CONTINUE
C
C           NOW RECOMPUTE THE PRESSURE, TEMPERATURE ETC.
C
            CALL E2PRMT(INODE,1)
200       CONTINUE

          RETURN
```

```
      END
```

G4SMOT

```
      SUBROUTINE G4SMOT(IT)

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'G2COMN.INC'
      DIMENSION DPLEFT(MEQNFL), DPRITE(MEQNFL)
      DIMENSION DPBOT (MEQNFL), DPTOP (MEQNFL)
      DATA SMALLP /1.E-10/, SMALLN /-1.E-10/
C     DATA SMALLP /1.E-3/, SMALLN /-1.E-3/

C*********************************************************************

C     THIS SUBROUTINE CORRECTS THE CONSERVATIVE VARIABLES AT A GIVEN
C     NODE 'INODE', IF THERE ARE OSCILLATIONS AT A NODE.  THE OSCILL-
C     ATIONS ARE DEFINED TO BE THE ONES WHICH CAUSE DISCONTINOUS
C     FIRST DIFFERENCES AT A NODE.  IT IS HOPED THAT SUCH A SITUATION
C     ONLY OCCURS AT A FEW NODES.

C*********************************************************************

      DO 100 INODE = 1, NNODG2
         NB1 = NEIBG2(1,INODE)
         NB2 = NEIBG2(2,INODE)
         NB3 = NEIBG2(3,INODE)
         NB4 = NEIBG2(4,INODE)

C        THE CORRECTION IS NOT APPLIED AT THE CORNER BOUNDARY CELLS

         IF (NB1 .EQ. 0 .AND. NB4 .EQ. 0) GOTO 100
         IF (NB2 .EQ. 0 .AND. NB3 .EQ. 0) GOTO 100
c        IF (NB1 .EQ. 0 .or. NB2 .EQ. 0 .or.
c    1       NB3 .EQ. 0 .or. NB4 .EQ. 0     ) GOTO 100

C        SETUP THE LEFT NEIGHBOUR CELL AND ITS NODE POINTER

         IF (NB1 .NE. 0) THEN
           NBLEFT = NB1
           IPLEFT = 8
         ELSE
           NBLEFT = NB4
           IPLEFT = 2
         ENDIF
C
C        FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
         IF (NB1 .EQ. NB4) THEN
           IPLEFT = 9
           IF (ICELG2(IPLEFT,NBLEFT) .EQ. 0) THEN
```

```
                INLFT1 = ICELG2(2,NBLEFT)
              -  INLFT2 = ICELG2(8,NBLEFT)
              -  XLEFT  = 0.5*(GEOMG2(1,INLFT1)+GEOMG2(1,INLFT2))
                 YLEFT  = 0.5*(GEOMG2(2,INLFT1)+GEOMG2(2,INLFT2))
                 DO 10 IQ = 1, NEQNFL
                   DPLEFT(IQ) = 0.5*(DPENG2(IQ,INLFT1)+DPENG2(IQ,INLFT2))
10               CONTINUE
                 GOTO 30
               ENDIF
             ENDIF

C            COMPUTE THE LEFT NODE, DISTANCES AND DP VARIABLES

             INLEFT = ICELG2(IPLEFT,NBLEFT)

             XLEFT  = GEOMG2(1,INLEFT)
             YLEFT  = GEOMG2(2,INLEFT)
             DO 20 IQ = 1, NEQNFL
                DPLEFT(IQ) = DPENG2(IQ,INLEFT)
20           CONTINUE

30           CONTINUE

C            SETUP THE RIGHT NEIGHBOUR CELL AND ITS NODE POINTER

             IF (NB2 .NE. 0) THEN
               NBRITE = NB2
               IPRITE = 6
             ELSE
               NBRITE = NB3
               IPRITE = 4
             ENDIF
C
C            FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
             IF (NB2 .EQ. NB3) THEN
               IPRITE = 5
               IF (ICELG2(IPRITE,NBRITE) .EQ. 0) THEN
                 INRIT1 = ICELG2(4,NBRITE)
                 INRIT2 = ICELG2(6,NBRITE)
                 XRITE  = 0.5*(GEOMG2(1,INRIT1)+GEOMG2(1,INRIT2))
                 YRITE  = 0.5*(GEOMG2(2,INRIT1)+GEOMG2(2,INRIT2))
                 DO 40 IQ = 1, NEQNFL
                   DPRITE(IQ) = 0.5*(DPENG2(IQ,INRIT1)+DPENG2(IQ,INRIT2))
40               CONTINUE
                 GOTO 60
               ENDIF
             ENDIF

C            COMPUTE THE RIGHT NODE, DISTANCES AND DP VARIABLES

             INRITE = ICELG2(IPRITE,NBRITE)
             XRITE  = GEOMG2(1,INRITE)
             YRITE  = GEOMG2(2,INRITE)

             DO 50 IQ = 1, NEQNFL
                DPRITE(IQ) = DPENG2(IQ,INRITE)
```

```
50          CONTINUE

60          CONTINUE

C           NOW CHECK FOR DENSITY DIFFERENCES ACROSS THE NODE

            DDLEFT = DPENG2(IT,INODE) - DPLEFT(IT)
            DDRITE = DPENG2(IT,INODE) - DPRITE(IT)
            IF (DDLEFT .GT. SMALLP .AND. DDRITE .GT. SMALLP) GOTO 70
            IF (DDLEFT .LT. SMALLN .AND. DDRITE .LT. SMALLN) GOTO 70
            GO TO 100

70          XNODE  = GEOMG2(1,INODE)
            YNODE  = GEOMG2(2,INODE)
            SNODE2 = (XNODE-XLEFT)**2 + (YNODE-YLEFT)**2
            SRITE2 = (XRITE-XLEFT)**2 + (YRITE-YLEFT)**2
            RATIO  = SQRT(SNODE2/SRITE2)
C
C           DO THE INTERPOLATION
C
            DO 80 IQ = 1, NEQNFL
               DPHERE = DPLEFT(IQ) + (DPRITE(IQ) -DPLEFT(IQ))*RATIO
               DPENG2(IQ,INODE) = 0.5*(DPHERE + DPENG2(IQ,INODE))
80          CONTINUE
C
C           NOW RECOMPUTE THE PRESSURE, TEMPERATURE ETC.
C
            CALL E2PRMT(INODE,1)
100       CONTINUE

C         NOW REPEAT THE WHOLE PROCESS FOR Y-AXIS

          DO 200 INODE = 1, NNODG2
             NB1 = NEIBG2(1,INODE)
             NB2 = NEIBG2(2,INODE)
             NB3 = NEIBG2(3,INODE)
             NB4 = NEIBG2(4,INODE)

C            THE CORRECTION IS NOT APPLIED AT THE CORNER BOUNDARY CELLS

c            IF (NB1 .EQ. 0 .AND. NB2 .EQ. 0) GOTO 200
c            IF (NB3 .EQ. 0 .AND. NB4 .EQ. 0) GOTO 200
             IF (NB1 .EQ. 0 .or. NB2 .EQ. 0 .or.
     1           NB3 .EQ. 0 .or. NB4 .EQ. 0     ) GOTO 200

C            SETUP THE BOTTOM NEIGHBOUR CELL AND ITS NODE POINTER

             IF (NB1 .NE. 0) THEN
               NBBOT = NB1
               IPBOT = 4
             ELSE
               NBBOT = NB2
               IPBOT = 2
             ENDIF
C
C            FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
```

```
            IF (NB1 .EQ. NB2) THEN
             IPBOT = 3
             IF (ICELG2(IPBOT,NBBOT) .EQ. 0) THEN
               INBOT1 = ICELG2(2,NBBOT)
               INBOT2 = ICELG2(4,NBBOT)
               XBOT  = 0.5*(GEOMG2(1,INBOT1)+GEOMG2(1,INBOT2))
               YBOT  = 0.5*(GEOMG2(2,INBOT1)+GEOMG2(2,INBOT2))
               DO 110 IQ = 1, NEQNFL
                 DPBOT(IQ) = 0.5*(DPENG2(IQ,INBOT1)+DPENG2(IQ,INBOT2))
110            CONTINUE
               GOTO 130
             ENDIF
            ENDIF

C           COMPUTE THE BOTTOM NODE, DISTANCES AND DP VARIABLES

            INBOT = ICELG2(IPBOT,NBBOT)

            XBOT  = GEOMG2(1,INBOT)
            YBOT  = GEOMG2(2,INBOT)
            DO 120 IQ = 1, NEQNFL
               DPBOT(IQ) = DPENG2(IQ,INBOT)
120         CONTINUE

130         CONTINUE

C           SETUP THE TOP NEIGHBOUR CELL AND ITS NODE POINTER

            IF (NB3 .NE. 0) THEN
              NBTOP = NB3
              IPTOP = 8
            ELSE
              NBTOP = NB4
              IPTOP = 6
            ENDIF
C
C           FOR SPATIAL INTERFACE, CORRECT THE NODE POINTER, ETC.
C
            IF (NB3 .EQ. NB4) THEN
              IPTOP = 7
              IF (ICELG2(IPTOP,NBTOP) .EQ. 0) THEN
                INTOP1 = ICELG2(6,NBTOP)
                INTOP2 = ICELG2(8,NBTOP)
                XTOP  = 0.5*(GEOMG2(1,INTOP1)+GEOMG2(1,INTOP2))
                YTOP  = 0.5*(GEOMG2(2,INTOP1)+GEOMG2(2,INTOP2))
                DO 140 IQ = 1, NEQNFL
                  DPTOP(IQ) = 0.5*(DPENG2(IQ,INTOP1)+DPENG2(IQ,INTOP2))
140             CONTINUE
                GOTO 160
              ENDIF
            ENDIF

C           COMPUTE THE TOP NODE, DISTANCES AND DP VARIABLES

            INTOP = ICELG2(IPTOP,NBTOP)
            XTOP  = GEOMG2(1,INTOP)
            YTOP  = GEOMG2(2,INTOP)
```

```
            DO 150 IQ = 1, NEQNFL
              DPTOP(IQ) = DPENG2(IQ,INTOP)
150         CONTINUE

160         CONTINUE

C           NOW CHECK FOR DENSITY DIFFERENCES ACROSS THE NODE

            DDBOT = DPENG2(IT,INODE) - DPBOT(IT)
            DDTOP = DPENG2(IT,INODE) - DPTOP(IT)
            IF (DDBOT .GT. SMALLP .AND. DDTOP .GT. SMALLP) GOTO 170
            IF (DDBOT .LT. SMALLN .AND. DDTOP .LT. SMALLN) GOTO 170
            GO TO 200

170         XNODE  = GEOMG2(1,INODE)
            YNODE  = GEOMG2(2,INODE)
            SNODE2 = (XNODE-XBOT)**2 + (YNODE-YBOT)**2
            STOP2 = (XTOP-XBOT)**2 + (YTOP-YBOT)**2
            RATIO  = SQRT(SNODE2/STOP2)
C
C           DO THE INTERPOLATION
C
            DO 180 IQ = 1, NEQNFL
               DPHERE = DPBOT(IQ) + (DPTOP(IQ) -DPBOT(IQ))*RATIO
               DPENG2(IQ,INODE) = 0.5*(DPHERE + DPENG2(IQ,INODE))
180         CONTINUE
C
C           NOW RECOMPUTE THE PRESSURE, TEMPERATURE ETC.
C
            CALL E2PRMT(INODE,1)
200      CONTINUE

         RETURN
         END
```

G2TIME

```
         SUBROUTINE G2TIME (TIME, IFIRST)

         INCLUDE '[.INC] PRECIS.INC/LIST'
         INCLUDE '[.INC] PARMV2.INC/LIST'
         INCLUDE '[.INC] E2COMN.INC/LIST'
         INCLUDE '[.INC] G2COMN.INC/LIST'
         INCLUDE '[.INC] IOCOMN.INC/LIST'
         INCLUDE '[.INC] TICOMN.INC/LIST'

         PARAMETER (MTIMA = 1000)
         DIMENSION TIMA(MTIMA)
         SAVE JC, NTIMA, TIMA

C*****************************************************************
```

```
C       THIS SUBROUTINE FIRST DETERMINES THE TIME-STATIONS WHERE THE
C       OUTPUT OF A RUN IS FIRST WRITTEN.  THESE TIME STATIONS COULD
C       BE AFTER EACH 'ITERATION' OR AFTER SPECIFIED TIME PERIODS.
C       IF OUTPUT AT ONLY THE FINAL TIME-PERIOD IS REQUIRED THEN THIS
C       ROUTINE IS NOT REALLY NEEDED.

C***********************************************************************

C       --------------
C       INITIALIZATION
C       --------------

        IF (IFIRST .EQ. 1) THEN

C           SET THE PARAMETER FOR THIS CASE (TO BE USED IN ONED_O)

C           KTIMTI = 1

C           INPUT THE NUMBER OF STATIONS WHERE RESULTS ARE NEEDED
C           INPUT -1 IF RESULTS ARE TO BE WRITTEN AFTER EACH ITERATION

           READ(JREADS,*) NTIMA
           IF (NTIMA .EQ. -1) THEN
              WRITE(JCARDS,1000) NTIMA
              TIMA(MTIMA) = -99.
              RETURN
           ENDIF

           JC = 1
           IF (KSRTE2 .EQ. 0 .OR. KTIMTI .EQ. 2) WRITE(JCARDS,1000) NTIMA

           DO 10 IT = 1, NTIMA
C              INPUT THE TIME FOR STATION #  I
              READ(JREADS,*) TIMA(IT)
10         CONTINUE
           TIMA(NTIMA+1) = 1000. + TIMXTI
           RETURN
        ENDIF

        IF (TIMNTI .GT. TIMA(JC) ) THEN
           JC = JC + 1
           RETURN
        ENDIF

C       --------------
C       EACH ITERATION
C       --------------

C       WRITE RESULTS FOR ALL THE ITERATIONS

        IF (TIMA(MTIMA) .EQ. -99.) THEN
           TIMA(JC) = TIME
           JC       = JC + 1
           WRITE(JCARDS,1000) JC,NNODG2,NEQNFL,NCELG2,TIME,TIMA(JC)
           DO 20 IC = 1, NCELG2
              WRITE(JCARDS,1100) IC, (ICELG2(J,IC), J = 2, 9)
```

```
20          CONTINUE
            DO 30 IN = 1, NNODG2
             - WRITE(JCARDS,1200) IN, (GEOMG2(J,IN), J = 1, 2     ),
     1                                  (DPENG2(J,IN), J = 1, NEQNFL)
30          CONTINUE
            RETURN
          ENDIF

C         -------------
C         SPECIFIC TIME
C         -------------

C         WRITE ONLY THE ITERATIONS AFTER SPECIFIC INTERVALS OF TIME

          IF (TIME .GE. TIMA(JC)) THEN
            WRITE(JCARDS,1000) JC,NNODG2,NEQNFL,NCELG2,TIME,TIMA(JC)
            JC       = JC + 1
            DO 40 IC = 1, NCELG2
              WRITE(JCARDS,1100) IC, (ICELG2(J,IC), J = 2, 9)
40          CONTINUE
            DO 50 IN = 1, NNODG2
              WRITE(JCARDS,1200) IN, (GEOMG2(J,IN), J = 1, 2     ),
     1                                  (DPENG2(J,IN), J = 1, NEQNFL)
50          CONTINUE
          ENDIF

C         -----------------
C         FORMAT STATEMENTS
C         -----------------

1000      FORMAT(4I5,2X,2G14.5)
1100      FORMAT(20I6)
1200      FORMAT(I5,2X,8G14.5)

          RETURN
          END
```

GETKY2

```
          SUBROUTINE GETKY2

          INCLUDE 'PRECIS.INC'
          INCLUDE 'PARMV2.INC'
          INCLUDE 'IOCOMN.INC'
          INCLUDE 'KYCOMN.INC'

          DIMENSION    IVAL(NIPAKY+NAPAKY)
          CHARACTER*7  KYWRDA(NAPAKY)     ,   KYWRDI(NIPAKY)
          CHARACTER    KEYFRM*15, KEYTRM*7,  NTYPE*4

          DATA KYWRDA/ 'SMAXE2=', 'SMINE2=',  'CFLNTI=', 'EPSLE2=',
     1                 'AMCHFL=', 'RHORFL=',  'TREFFL=', 'TREFCH=',
     2                 'PRESFL=', 'PRESCH=',  'DISTFL=', 'TEMP1C=',
```

```
     3              'TEMP2C=', 'TEMP3C=',  'ALPHA2=', 'BETAA2=',
     4      -       'GAMMA2=', 'DELTA2=',  'PRINTO=', 'TIMXTI=',
     5      -       'PBPIFR=', 'EPS1TI=',  'EPSOTI=', 'TIMNTI=',
     6              'TRIGCH=', 'ERRMIN=',  'ERRMTI=', 'EPS1MN=',
     7              'EPS1MX=', 'RHORFR=',  'UCOMFR=', 'VCOMFR=',
     8              'PRESFR=', 'FCTRTI=',  'DTCNTI=', 'RREYE2=',
     9              'RPRNE2=', 'RSCHE2=',  'OMEGE2=', 'GFACE2=',
     *              'CFLXTI=', '      ='/

        DATA KYWRDI/ 'NREACH=', 'NSPECH=',  'KROGER=', 'KORDER=',
     1              'MITRE2=', 'KSRTE2=',  'NGIVTI=', 'METHA2=',
     2              'JREADS=', 'KTIMTI=',  'K1ADA2=', 'K2ADA2=',
     3              'KDEBUG=', 'IDBGE2=',  'IDBGA2=', 'MTHRA2=',
     4              'KFACTI=', 'NINRCH=',  'KDPENI=', 'KPLTA2=',
     5              'IDBGFR=', 'NXTDA2=',  'MALVG2=', 'KADPTI=',
     6              'KONVE2=', 'MITRA2=',  'MITRPS=', 'KHAFEZ=',
     7              'KEQNE2=', 'KMERA2=',  'KCHKA2=', 'MITEPS=',
     8              'IMPLTI=', 'IDBGTI=',  'KPERFR=', 'MCYCFR=',
     9              'IDBGG2=', 'IADDH2=',  'KDIFTI=', 'KBLOCK=',
     *              '      =', '      ='/

C***********************************************************************

C       THIS SUBROUTINE READS THE INPUT RECORDS, FINDS THE CORRESPONDING
C       KEYWORDS AND ASSIGNS VALUES TO THE RESPECTIVE VARIABLES.  THE
C       INPUT RECORDS CAN BE IN ANY ORDER, HOWEVER THE FOLLOWING CONVENTIONS
C       MUST BE OBSERVED :
C
C         1.  ALL THE KEYWORDS MUST BE EXACTLY SIX BYTES LONG, IF THERE
C             ARE BLANKS IN THE KEYWORD THEN THOSE BLANKS MUST BE IN THE
C             TRAILING BYTES
C
C         2.  THE SEVENTH BYTE MUST BE THE ASSIGN SYMBOL '='
C
C         3.  IF THE KEYWORD IS FOR AN INTEGER THEN IT SHOULD BE
C             WRITTEN IN FORMAT I5, I.E., THE INTEGER VALUE BE
C             SPECIFIED IN COLUMNS 8-12 FOLLOWING THE FIRST SEVEN
C             RESERVED FOR THE KEYWORD NAME
C
C         4.  IF THE KEYWORD IS REAL IT MUST BE WRITTEN IN THE G FORMAT
C             FOLLOWING COLUMN 7

C***********************************************************************
C
C        ------------------
C        KEYWORD ASSIGNMENTS
C        ------------------
C                        KYWRDA   DEFINED BY APASKY
C              1.   'SMAXE2='         2.    'SMINE2='
C              3.   'CFLNTI='         4.    'EPSLE2='
C              5.   'AMCHFL='         6.    'RHORFL='
C              7.   'TREFFL='         8.    'TREFCH='
C              9.   'PRESFL='         10.   'PRESCH='
C              11.  'DISTFL='         12.   'TEMP1C='
C              13.  'TEMP2C='         14.   'TEMP3C='
C              15.  'ALPHA2='         16.   'BETAA2='
C              17.  'GAMMA2='         18.   'DELTA2='
```

```
C              19.  'PRINTO='          20.   'TIMXTI='
C              21.  'PBPIFR='          22.   'EPS1TI='
C              23.  'EPSOTI='          24.   'TIMNTI='
C              25.  'TRIGCH='          26.   'ERRMIN='
C              27.  'ERRMTI='          28.   'EPS1MN='
C              29.  'EPS1MX='          30.   'RHORFR='
C              31.  'UCOMFR='          32.   'VCOMFR='
C              33.  'PRESFR='          34.   'FCTRTI='
C              35.  'DTCNTI='          36.   'RREYE2='
C              37.  'RPRNE2='          38.   'RSCHE2='
C              39.  'OMEGE2='          40.   'GFACE2='
C              41.  'CFLXTI='          42.   '      ='
C
C
C                     KYWRDI   DEFINED BY IPASKY
C              1.   'NREACH='          2.    'NSPECH='
C              3.   'KROGER='          4.    'KORDER='
C              5.   'MITRE2='          6.    'KSRTE2='
C              7.   'NGIVTI='          8.    'METHA2='
C              9.   'JREADS='          10.   'KTIMTI='
C              11.  'K1ADA2='          12.   'K2ADA2='
C              13.  'KDEBUG='          14.   'IDBGE2='
C              15.  'IDBGA2='          16.   'MTHRA2='
C              17.  'KFACTI='          18.   'NINRCH='
C              19.  'KDPENI='          20.   'KPLTA2='
C              21.  'IDBGFR='          22.   'NXTDA2='
C              23.  'MALVG2='          24.   'KADPTI='
C              25.  'KONVE2='          26.   'MITRA2='
C              27.  'MITRPS='          28.   'KHAFEZ='
C              29.  'KEQNE2='          30.   'KMERA2='
C              31.  'KCHKA2='          32.   'MITEPS='
C              33.  'IMPLTI='          34.   'IDBGTI='
C              35.  'KPERFR='          36.   'MCYCFR='
C              37.  'IDBGG2='          38.   'IADDH2='
C              39.  'KDIFTI='          40.   'KBLOCK='
C              41.  '      ='          42.   '      ='
C
C
C     ------------------------------------------
C     SYNOPSIS OF LETTERS IN THE NAME ASSIGNMENTS
C     ------------------------------------------
C
C     FIRST LETTER
C     J    :  STANDS FOR UNITS
C     M    :  MAXIMUM VALUE
C     N    :  NUMBER OF
C     K    :  DECISION PARAMETERS
C
C     LAST TWO LETTERS
C     CH   :  NUMBERS TO BE INITIALIZED IN CHINIT (CHCOMN COMMON BLOCK)
C     FL   :  NUMBERS TO BE INITIALIZED IN FLINIT (FLCOMN COMMON BLOCK)

      NVALUE = 0
      KDEBUG = 1
C
C     INITIALIZE ALL THE KEYS TO BE MARKED TO HAVE DEFAULT VALUES, I.E,
C     MARIKY=0; IF A KEY IS FOUND HERE IT WOULD BE MARKED TO HAVE NON-
```

```
C       DEFAULT VALUES
C
        DO 5 IKEY = 1, NIPAKY
           MARIKY(IKEY) = 0
5       CONTINUE

        DO 6 IKEY = 1, NAPAKY
           MARAKY(IKEY) = 0
6       CONTINUE

C       MAKE SURE THAT THE FUNCTION ICHAR IS DEFINED ON ALL THE COMPUTERS

        NN = ICHAR('N')
        II = ICHAR('I')

10      READ (JREADI,1000,END=50) KEYTRM, KEYFRM

        BACKSPACE (JREADI)

C       CHECK IF THE KEYTRM IS AN INTEGER

        N1 = ICHAR(KEYTRM(1:1))

        IF (N1 .GE. II .AND. N1 .LE. NN) THEN
            NTYPE = 'INTE'
            READ (JREADI,1100) KEYTRM, IVALKY
        ELSE
            NTYPE = 'REAL'
            READ (JREADI,1200) KEYTRM, AVALKY
        ENDIF

C       NVALUE COUNTS THE NUMBER OF RECORDS CONTAINING THE STANDARD KEYWORDS

        NVALUE = NVALUE + 1

        IF (NTYPE .EQ. 'REAL') GOTO 30

C       THE KEYWORD CORRESPONDS TO AN INTEGER VARIABLE

        DO 20 J = 1, NIPAKY
            IF (KEYTRM .EQ. KYWRDI(J)) THEN
               IPASKY(J)    = IVALKY
               IVAL(NVALUE) = J
               MARIKY(J)    = 1
               GOTO 10
            ENDIF
20      CONTINUE

        ZER1 = NVALUE
        ZER2 = IVALKY

        CALL ERRORM (1,'GETKY2','REC # ',ZER1,KEYTRM,ZER2,JPRINT,
     1                 'KEYWORD NOT FOUND')

C       THE KEYWORD CORRESPONDS TO A REAL VARIABLE

30      DO 40 J = 1, NAPAKY
```

```
            IF (KEYTRM .EQ. KYWRDA(J)) THEN
             - APASKY(J)    = AVALKY
             - IVAL(NVALUE) = J + NIPAKY
               MARAKY(J)    = 1
               GOTO 10
            ENDIF
40       CONTINUE

         ZER1 = NVALUE
         ZER2 = AVALKY

         CALL ERRORM (1,'GETKY2','REC # ',ZER1,KEYTRM,ZER2,JPRINT,
     1                'KEYWORD NOT FOUND')


50       KDEBUG = IPASKY(13)
         IF (KDEBUG .EQ. 1 .OR. KDEBUG .GT. 999) THEN
            WRITE(JDEBUG,1300)
            WRITE(JDEBUG,1400)
            WRITE(JDEBUG,1500)
            WRITE(JDEBUG,1600)
            DO 90 J = 1, NVALUE
               IV = IVAL(J)
               IF (IV .LE. NIPAKY) THEN
                  WRITE(JDEBUG,1700) J,IV,KYWRDI(IV),IPASKY(IV)
               ELSE
                  IV = IV - NIPAKY
                  WRITE(JDEBUG,1800) J,IV,KYWRDA(IV),APASKY(IV)
               ENDIF
90          CONTINUE
         ENDIF

C        -----------------
C        FORMAT STATEMENTS
C        -----------------

1000     FORMAT(A7,A15)
1100     FORMAT(A7,I5 )
1200     FORMAT(A7,G  )
C        UNCOMMENT THE FOLLOWING LINE FOR CYBER COMPUTER
C1200    FORMAT(A7,G14.5)
1300     FORMAT(//10X,'----------------------' )
1400     FORMAT(  10X,'DEBUG PRINT FROM GETKY2' )
1500     FORMAT(  10X,'----------------------'/)
1600     FORMAT(   2X,'REC #',5X,'KEYWORD #',4X,'KEYWORD',
     1           11X,'KEYWORD VALUE'/)
1700     FORMAT(I5,5X,I5,10X,A7,13X,I7   )
1800     FORMAT(I5,5X,I5,10X,A7,13X,G14.5)

         RETURN
         END
```

H2EMBD

```
      SUBROUTINE H2EMBD

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] H2COMN.INC/LIST'
      INCLUDE '[.INC] HEXCOD.INC     '
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] PRCOMN.INC/LIST'

C*********************************************************************
C
C     THIS SUBROUTINE ADDS EMBEDDED CELLS ACROSS THE FUEL ADDITION
C     PLANE.  THESE CELLS ARE PERMANENTLY DIVIDED AND NEVER ALLOWED
C     TO COLLAPSE AGAIN.  THE LEVELS OF EMBEDDING ACROSS THE PLANE
C     OF INJECTION EQUALS THE CURRENT MAXIMUM EMBEDDING LEVEL
C
C*********************************************************************
C
      DO 180 ILEVEL = 1, MALVG2

        INODE  = IBASH2
        NBTYP1 = 4
        INTYP1 = 6
        NBTYP2 = 3
        INTYP2 = 8
        NB1    = NEIBG2(4,INODE)
        NB2    = NEIBG2(3,INODE)
C
C       ERROR CONDITIONS
C
        IF (NB1 .EQ. 0) THEN
         ZER1 = ISTART
         ZER2 = NB1
         CALL ERRORM (46,'H2INIT','ISTART',ZER1,'NB1  ',ZER2,JPRINT,
     1            'ERROR IN NEIGHBOUR CELLS OF STARTING POINT')
        ENDIF
C
        IF (NB2 .EQ. 0) THEN
         ZER1 = ISTART
         ZER2 = NB2
         CALL ERRORM (46,'H2INIT','ISTART',ZER1,'NB2   ',ZER2,JPRINT,
     1            'ERROR IN NEIGHBOUR CELLS OF STARTING POINT')
        ENDIF
C
C       NOW MARCH IN THE APPROPRIATE DIRECTION
C
C       SAVE THE NODE WHERE THE FUEL IS TO INJECTED
170     NBNXT1        = NEIBG2(NBTYP1,INODE)
        NBNXT2        = NEIBG2(NBTYP2,INODE)
        IF (NBNXT1 .EQ. 0 .OR. NBNXT2 .EQ. 0) GOTO 180
C       MARK THE CELL WHERE FUEL IS TO ADDED
        KAUXG2(NBNXT1) = IOR(KAUXG2(NBNXT1),KL2000)
```

```
          KAUXG2(NBNXT2) = IOR(KAUXG2(NBNXT2),KL2000)
          IWARN          = 0
          CALL  G2DIVU (NBNXT1,IWARN)
          CALL  G2DIVU (NBNXT2,IWARN)
          INODE          = ICELG2(INTYP1,NBNXT1)
          GO TO 170

180       CONTINUE
C
C       ADJUST THE ARRAY PERTAINING TO THE INJECTION PLANE
C
        CALL H2INIT

        RETURN
        END
```

H2FLOT

```
        SUBROUTINE H2FLOT

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
C
C**********************************************************************

C       THIS SUBROUTINE ALLOWS THE ASSIGNMENT OF VALUES TO THE NODES AT
C       THE RIGHT OF A GIVEN NODE TO THE VALUES OF THE GIVEN NODE ITSELF.

C**********************************************************************
C
C       READ THE FOLLOWING QUANTITIES
C            NBASE   : TOTAL NUMBER OF BASE NODES
C            IBASE   : THE BASE NODE ITSELF
C
        READ (JREADS,*) NBASE

        DO 20 II = 1, NBASE
           READ (JREADS,*) IBASE
           INODE  = IBASE
           NBTYPE = 0
           NB1    = NEIBG2(2,INODE)
           NB2    = NEIBG2(3,INODE)

           IF (NB1 .NE. 0) THEN
              NBTYPE = 2
              INTYPE = 6
           ELSEIF (NB2 .NE. 0) THEN
              NBTYPE = 3
              INTYPE = 4
           ENDIF
C
```

```
C          ERROR CONDITION
C          -
           IF (NBTYPE .EQ. 0) THEN
              ZER1 = ISTART
              ZER2 = NBTYPE
              CALL ERRORM (46,'H2FLOT','ISTART',ZER1,'NBTYPE',ZER2,
     1             JPRINT,'ERROR IN NEIGHBOUR CELLS OF STARTING POINT')
           ENDIF
C
C          NOW MARCH IN THE APPROPRIATE DIRECTION
C
C          FIND THE NEXT CELL ON TOP OF THE NODE UNDER CONSIDERATION
10         NBNEXT = NEIBG2(NBTYPE,INODE)

C          SEE IF YOU HAVE REACHED THE RIGHT-MOST BOUNDARY SURFACE
           IF (NBNEXT .EQ. 0) GOTO 20

C          FIND THE NEXT NODE TO THE RIGHT
           INODE  = ICELG2(INTYPE,NBNEXT)

C          ASSIGN THE VALUES

           DO 15 IQ = 1, NEQNFL
              DPENG2(IQ,INODE) = DPENG2(IQ,IBASE)
15         CONTINUE

           PRESG2(INODE) = PRESG2(IBASE)
           TEMPG2(INODE) = TEMPG2(IBASE)

           GO TO 10

20      CONTINUE

        RETURN
        END
```

H2INIT

```
        SUBROUTINE H2INIT

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] CHCOMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
        INCLUDE '[.INC] H2COMN.INC/LIST'
        INCLUDE '[.INC] HEXCOD.INC      '
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        INCLUDE '[.INC] PRCOMN.INC/LIST'

C***********************************************************************
C
C       THIS SUBROUTINE INITIALIZES THE FUEL ADDITION PLANE (VERTICAL)
C       FOR A GIVEN EQUIVALENCE RATIO AND A BASE NODE; SEE E2SCHO AND
```

```
C       H2EMBD FOR INITIALIZING THROUGH THIS SUBROUTINE
C          -
C*************************************************************************
C
        INODE  = IBASH2
        NBTYPE = 0
        NB1    = NEIBG2(4,INODE)
        NB2    = NEIBG2(3,INODE)
        IF (NB1 .NE. 0) THEN
           NBTYPE = 4
           INTYPE = 6
        ELSEIF (NB2 .NE. 0) THEN
           NBTYPE = 3
           INTYPE = 8
        ENDIF
C
C       ERROR CONDITION
C
        IF (NBTYPE .EQ. 0) THEN
           ZER1 = ISTART
           ZER2 = NBTYPE
           CALL ERRORM (46,'H2INIT','ISTART',ZER1,'NBTYPE',ZER2,JPRINT,
     1            'ERROR IN NEIGHBOUR CELLS OF STARTING POINT')
        ENDIF
C
C       NOW MARCH IN THE APPROPRIATE DIRECTION
C
        KOUNT  = 0
        NCELH2 = 0
170     KOUNT     = KOUNT + 1
C       SAVE THE NODE WHERE THE FUEL IS TO INJECTED
        NODEH2(KOUNT) = INODE
        NBNEXT        = NEIBG2(NBTYPE,INODE)
        IF (NBNEXT .EQ. 0) GOTO 180
C       MARK THE CELL WHERE FUEL IS TO ADDED
        KAUXG2(NBNEXT) = IOR(KAUXG2(NBNEXT),KL1000)
        NCELH2 = NCELH2 + 1
        ICELH2(NCELH2) = NBNEXT
        INODE          = ICELG2(INTYPE,NBNEXT)
        GO TO 170

C       SAVE THE TOTAL NUMBER OF FUEL INJECTION POINTS
180     NUMDH2  = KOUNT

C       FOR ROGERS AND CHINITZ MODEL THE NUMBER OF EQUATIONS MUST
C       BE ADJUSTED

        IF (KROGER .EQ. 1 .AND. NINRCH .GT. 0) THEN
           DO 200 INODE = 1, NNODG2

C             COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
              SUMY   = 0.
              YUPPER = 1. - YNRTCH
              RHORPR = DPENG2(1,INODE)

              DO 190 IS = 1, NEQSCH
                 JS         = NEQBAS + IS
```

```
                YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
              -  IF (YSPEPR(IS) .LT. 0.) THEN
              -    YSPEPR(IS)       = 0.
                   DPENG2(JS,INODE) = 0.
                ENDIF
                IF (YSPEPR(IS) .GT. YUPPER) THEN
                  YSPEPR(IS)       = YUPPER
                  DPENG2(JS,INODE) = YUPPER*RHORPR
                ENDIF
                SUMY       = SUMY + YSPEPR(IS)
190           CONTINUE
C             THE FOLLOWING IS FOR SPECIES 4 = NEQSCH+1
              YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
              IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C             ADJUST THE NEWLY DEFINED VARIABLE AT THIS NODE
              DPENG2(NEQNFL+1,INODE) = RHORPR*YSPEPR(NEQSCH+1)
200         CONTINUE
C           NOW ADJUST THE NUMBER OF EQUATIONS
            YNRTCH = 0.
            NEQNFL = NEQNFL + 1
            NEQSCH = NEQSCH + 1
            NINRCH = NINRCH - 1
          ENDIF
C
          RETURN
          END
```

H3INIT

```
          SUBROUTINE H3INIT
C
          INCLUDE 'PRECIS.INC'
          INCLUDE 'PARMV2.INC'
          INCLUDE 'CHCOMN.INC'
          INCLUDE 'E2COMN.INC'
          INCLUDE 'FLCOMN.INC'
          INCLUDE 'FRCOMN.INC'
          INCLUDE 'G2COMN.INC'
C         INCLUDE 'H2COMN.INC'
          INCLUDE 'HEXCOD.INC'
          INCLUDE 'IOCOMN.INC'
          INCLUDE 'KYCOMN.INC'
          INCLUDE 'PRCOMN.INC'
C
C*****************************************************************

C         THIS SUBROUTINE INITIALIZES THE DEPENDENT VARIABLES FOR FUEL
C         INJECTION AS WALL POINTS AS A MIXTURE OF FUEL AND AIR.
C         THE VALUES NEEDED AT THE WALL POINTS ARE THE PROPERTIES OF THIS
C         MIXTURE, I.E., TEMPERATURE, PRESSURE, MACH NO., EQUIVALENCE
C         RATIO, AND THE ANGLE OF INJECTION.  ALSO NEEDED IS THE TOTAL
C         NUMBER OF WALL CELLS AND THE ACTUAL CELL NUMBERS.
```

```
C*********************************************************************
C          -
C       IF (KROGER .NE. 1) RETURN
C
C       READ THE FOLLOWING FUEL QUANTITIES
C            TEMPEF  : FUEL TEMPERATURE IN DEGREE K
C            PRESSF  : FUEL PRESSURE IN PASCALS
C            AMACHF  : FUEL MACH NUMBER
C            EQUIVF  : EQUIVALENCE RATIO
C            ANGLEF  : ANGLE OF INJECTION IN DEGREES
C            NCELLF  : NUMBER OF CELLS WITH FUEL INJECTION
C            ICELL   : CELLS WHERE FUEL IS INJECTED
C            IF EQUIVF > 100 THEN ONLY FUEL IS ADDED AT THE INJECTORS
C
       READ (JREADS,*) TEMPEF
       READ (JREADS,*) PRESSF
       READ (JREADS,*) AMACHF
       READ (JREADS,*) EQUIVF
       READ (JREADS,*) ANGLEF
       READ (JREADS,*) NCELLF
C
C      COMPUTE THE ANGLE IN RADIANS
       ANGLEF = ANGLEF*3.141592654/180.

C      DETERMINE THE MASS FRACTION OF H2 BASED ON EQUIVALENCE RATIO
C      AND OTHER MASS FRACTIONS
C      YH2 = 2 PHI M_H2 / (M_O2 + 3.76 M_N2 + 2 PHI M_H2)

       YSPEPR(2) = 0.
       YSPEPR(4) = 0.
       IF (EQUIVF .GT. 100.) THEN
          YSPEPR(1)    = 0.
          YSPEPR(3)    = 1.
          YSPEPR(5)    = 0.
       ELSE
          YSPEPR(1)    = 7.93626/(EQUIVF+34.048)
          YSPEPR(3)    = EQUIVF/(EQUIVF+34.048)
          YSPEPR(5)    = 1. - YSPEPR(1) - YSPEPR(3)
       ENDIF
C
C      DETERMINE THE MOLECULAR MASS AND OTHER QUANTITIES FOR THIS MIXTURE
C
       DO 5 IS = 1, NSPECH
           SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
           SYSHFE = SYSHFE + YSPEPR(IS)*FMHTCH(IS)
           SYSCPE = SYSCPE + YSPEPR(IS)*SPCPCH(IS)
           BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
5      CONTINUE

       UGASCO = UGASFL*SYSBMS

C      DETERMINE THE DIMENSIONLESS DENSITY OF THE FUEL MIXTURE
       RHOF   = PRESSF/(UGASCO*TEMPEF*RHORFL)

C      DETERMINE THE DIMENSIONLESS PRESSURE OF THE FUEL MIXTURE
       PRESSF = PRESSF/PRESFL
C
```

```
C       DETERMINE GAMMA FOR THIS MIXTURE
        BIGAMT = BIGAM*TEMPEF
        SYSCVE = SYSCPE + BIGAMT - UGASFL*SYSBMS
        GAMMAE = (SYSCPE + BIGAMT)/SYSCVE

C       DETERMINE THE OVERALL DIMENSIONLESS VELOCITY OF THE FUEL
        VELOF  = AMACHF*SQRT(GAMMAE*PRESSF/RHOF)
        UCOMPF = VELOF*COS(ANGLEF)
        VCOMPF = VELOF*SIN(ANGLEF)
        VEL02I = UCOMPF*UCOMPF + VCOMPF*VCOMPF

C       DETERMINE THE ENERGY TERM
        BEE    = SYSHFE + (TEMPEF-TREFCH)*SYSCPE - UGASFL*TEMPEF*SYSBMS
     1                        + 0.5*(TEMPEF*TEMPEF-TREFCH*TREFCH)*BIGAM
        BEE    = BEE/FMREFL + 0.5*VEL02I
C
        TEMPEF = TEMPEF/TREFFL

        DO 30 JCELL = 1, NCELLF
C          READ THE CELL NUMBER FOR THIS VALUE
           READ (JREADS,*) ICELLF
           KX    = KAUXG2(ICELLF)
           IEDGE = IAND(KX,KL000F)
           IF (IEDGE .EQ. KL0003) THEN
              VHERE  = VCOMPF
              INODE1 = ICELG2(2,ICELLF)
              INODE2 = ICELG2(4,ICELLF)
           ELSE IF (IEDGE .EQ. KL000C) THEN
              VHERE  =-VCOMPF
              INODE1 = ICELG2(6,ICELLF)
              INODE2 = ICELG2(8,ICELLF)
           ELSE
              GOTO 30
           ENDIF

C          SET THE DEPENDENT VARIABLES
           DPENG2(1,INODE1) = RHOF
           DPENG2(1,INODE2) = RHOF
           DPENG2(2,INODE1) = RHOF*UCOMPF
           DPENG2(2,INODE2) = RHOF*UCOMPF
           DPENG2(3,INODE1) = RHOF*VHERE
           DPENG2(3,INODE2) = RHOF*VHERE
           DPENG2(4,INODE1) = BEE*RHOF
           DPENG2(4,INODE2) = BEE*RHOF

           DO 10 JS = NEQBAS+1, NEQNFL
              IS             = JS - NEQBAS
              DPENG2(JS,INODE1) = RHOF*YSPEPR(IS)
              DPENG2(JS,INODE2) = RHOF*YSPEPR(IS)
10         CONTINUE

           PRESG2(INODE1) = PRESSF
           TEMPG2(INODE1) = TEMPEF
           PRESG2(INODE2) = PRESSF
           TEMPG2(INODE2) = TEMPEF

C          SET THE BOUNDARY CONDITION POINTER
```

```
            DO 20 JS = 1, NBNDG2
             - IF (IBNDG2(1,JS) .EQ. INODE1) IBNDG2(5,JS) = 2
             - IF (IBNDG2(1,JS) .EQ. INODE2) IBNDG2(5,JS) = 2
20          CONTINUE

30        CONTINUE
C
C         FOR ROGERS AND CHINITZ MODEL THE NUMBER OF EQUATIONS MUST
C         BE ADJUSTED

          IF (KROGER .EQ. 1 .AND. NINRCH .GT. 0) THEN
             DO 200 INODE = 1, NNODG2

C               COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
                SUMY   = 0.
                YUPPER = 1. - YNRTCH
                RHORPR = DPENG2(1,INODE)

                DO 190 IS = 1, NEQSCH
                   JS         = NEQBAS + IS
                   YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
                   IF (YSPEPR(IS) .LT. 0.) THEN
                     YSPEPR(IS)       = 0.
                     DPENG2(JS,INODE) = 0.
                   ENDIF
                   IF (YSPEPR(IS) .GT. 1.) THEN
                     YSPEPR(IS)       = 1.
                     DPENG2(JS,INODE) = YUPPER*RHORPR
                   ENDIF
                   SUMY       = SUMY + YSPEPR(IS)
190             CONTINUE
C               THE FOLLOWING IS FOR SPECIES 4 = NEQSCH+1
                YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
                IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C               ADJUST THE NEWLY DEFINED VARIABLE AT THIS NODE
                DPENG2(NEQNFL+1,INODE) = RHORPR*YSPEPR(NEQSCH+1)
200          CONTINUE

C            NOW ADJUST THE NUMBER OF EQUATIONS
             YNRTCH = 0.
             NEQNFL = NEQNFL + 1
             NEQSCH = NEQSCH + 1
             NINRCH = NINRCH - 1
          ENDIF

          RETURN
          END
```

H2MIXT

```
          SUBROUTINE H2MIXT(ITGL)

          INCLUDE '[.INC] PRECIS.INC/LIST'
```

```
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] e2COMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] H2COMN.INC/LIST'
      INCLUDE '[.INC] PRCOMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      DIMENSION YSPEH2(MEQNFL), ENTLH2(MEQNFL), KODEH2(MUMDH2)

C*********************************************************************

C     THIS SUBROUTINE INJECTS THE FUEL AT THE PREVIOUSLY GIVEN
C     LOCATIONS (STORED IN NODEH2)

C*********************************************************************
C
C     IF (KROGER .NE. 1) RETURN
C
C     DETERMINE THE MASS FRACTION OF H2 BASED ON EQUIVALENCE RATIO
      YH2    = PHIEH2/(PHIEH2+34.048)

      DO 50 JNODE = 1, NUMDH2
C
C       DETERMINE THE ACTUAL NODE OF INJECTION
C
        INODE  = NODEH2(JNODE)
        IF (CHNGE2(1,INODE) .EQ. 0.) GOTO 50
C
C       DETERMINE THE PRIMITIVE VARIABLES BEFORE FUEL INJECTION
C       CORRECTION
C
        RHORPR = DPENG2(1,INODE)
        UCOMPR = DPENG2(2,INODE)/RHORPR
        VCOMPR = DPENG2(3,INODE)/RHORPR
        BEPSPR = DPENG2(4,INODE)
        VELO2U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C       COMPUTE THE DIMENSIONAL QUANTITIES
C
        TEMPPR = TEMPG2(INODE)*TREFFL

C       COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
        SUMY   = 0.
        DO 10 IS = 1, NEQSCH
          JS         = NEQBAS + IS
          YSPEPR(IS) = DPENG2(JS,INODE)/DPENG2(1,INODE)
          IF (YSPEPR(IS) .LT. 0.) THEN
            YSPEPR(IS)       = 0.
            DPENG2(JS,INODE) = 0.
          ENDIF
          SUMY       = SUMY + YSPEPR(IS)
10      CONTINUE

        YSPEPR(NEQSCH+1) = 1. - SUMY
        IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C       DETERMINE THE CURRENT MASS FRACTION OF THE FUEL
```

```
      YH2P = YSPEPR(3)
C
C     DETERMINE THE FUEL QUANTITIES
C
      RHOF = RHORPR*(YH2-YH2P)/(1.-YH2)
      PF   = RHOF*RHORFL*UGASFL*RAMWCH(3)*TEMPPR/PRESFL
C
      AMASSX = RHORPR*UCOMPR + RHOF*UCOMPR
      FORCEX = RHORPR*UCOMPR*UCOMPR + RHOF*UCOMPR*UCOMPR +
     1         PRESG2(INODE) + PF

C     DETERMINE THE FACTOR BY WHICH THE MASS FRACTIONS MUST BE
C     ADJUSTED
      YRAT = (1.-YH2)/(1.-YH2P)
C
C     COMPUTE THE NEW MASS FRACTIONS, MOLECULAR MASSES AND THE
C     ENTHALPY OF EACH SPECIES
C
      SYSBMO = 0.
      SYSBMN = 0.
      TENTHI = 0.
      YSPEH2(3) = YH2
      HM        = 0.

      DO 20 IS = 1, NSPECH
         IF (IS .NE. 3) YSPEH2(IS) = YSPEPR(IS)*YRAT
         SYSBMO = SYSBMO + YSPEPR(IS)*RAMWCH(IS)
         SYSBMN = SYSBMN + YSPEH2(IS)*RAMWCH(IS)
         ENTLH2(IS) = FMHTCH(IS) +
     1         SPCPCH(IS)*(TEMPPR-TREFCH) +
     2       0.5*SPBSCH(IS)*(TEMPPR**2-TREFCH**2)
         TENTHI = TENTHI + ENTLH2(IS)*YSPEPR(IS)
         HM     = HM + FMHTCH(IS)*YSPEH2(IS)
20    CONTINUE

      TENTHI = TENTHI/FMREFL + 0.5*VEL02U
      TENTHF = ENTLH2(3)/FMREFL + 0.5*VEL02U
      ENRGYX = UCOMPR*RHORPR*TENTHI + UCOMPR*RHOF*TENTHF

C     MOLECULAR MASSES
      AMASOL = 1./SYSBMO
      AMASNW = 1./SYSBMN

      UGASM  = UGASFL*SYSBMN
      TMRAT  = PRESFL/(RHORFL*UGASM)

C     INITIAL GUESS FOR DENSITY
      RHOM = RHORPR + RHOF
      VM   = VCOMPR

1001  UM   = AMASSX/RHOM
      PM   = FORCEX - UM*AMASSX
      TMD  = PM*TMRAT/RHOM
      VEL02M = 0.5*(UM*UM+VM*VM)
C
C     DETERMINE THE NEW MIXTURE ENTHALPY
```

```
          SYSENT = 0.
          DO 30 IS = 1, NSPECH
            ENTLH2(IS) = SPCPCH(IS)*(TMD-TREFCH) +
     2        0.5*SPBSCH(IS)*(TMD**2-TREFCH**2)
            SYSENT = SYSENT + YSPEH2(IS)*ENTLH2(IS)
30        CONTINUE

          ENTHM = (HM+SYSENT)/FMREFL
          RHOMN = ENRGYX/(enthm+VELO2M)/UM
C         COMPA = ABS(RHOM-RHOMN)
          RHOM = RHOMN
C         IF (COMPA .GT. 1.E-4) GOTO 1001

          BENW = (ENTHM+VELO2M)*RHOM - pm
C
C         ADJUST THE DEPENDENT VARIABLES
C
          DPENG2(1,INODE) = RHOM
          DPENG2(2,INODE) = RHOM*UM
          DPENG2(3,INODE) = RHOM*VM
          DPENG2(4,INODE) = BENW
          DO 40 IS = 1, NEQSCH
            JS          = NEQBAS + IS
            DPENG2(JS,INODE) = RHOM*YSPEH2(IS)
40        CONTINUE

50      CONTINUE

        CALL H2SOLF (ITGL)
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGCH .NE. 19 .AND. IDBGCH .LT. 1000) RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) RHORPR, RHOM, BEPSPR, BENW,
     1                     AMASOL, AMASNW, UCOMPR, VCOMPR
        DO 60 IS = 1, NSPECH
          WRITE(JDEBUG,1400) YSPEPR(IS), YSPEH2(IS)
60        CONTINUE
        WRITE(JDEBUG,1500) IADDH2, NUMDH2, PHIEH2
        WRITE(JDEBUG,1600)
        WRITE(JDEBUG,1700) (NODEH2(INODE), INODE=1,NUMDH2)
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM H2MIXT' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(15X,'OLD VALUES',6X,'NEW VALUES'/
     1         5X,'DENSITY  ',5X,2G14.6/
     2         5X,'ENERGY   ',5X,2G14.6/
     3         5X,'MOL MASS ',5X,2G14.6/
```

```
     4          5X,'VELOCITY ',5X,2G14.6,5X,'SAME U & V COMPONENTS'/)
1400    FORMAT( 5X,'MASS FRAC',5X,2G14.6)
1500    FORMAT(/5X,'IAADH2 =',I5,10X,'NUMDH2 =',I5,
     1         10X,'PHIEH2 =',G14.6)
1600    FORMAT (/10X,'FUEL INJECTION POINTS')
1700    FORMAT (10I7)

        RETURN
        END
```

H3MIXT

```
        SUBROUTINE H2MIXT

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] CHCOMN.INC/LIST'
        INCLUDE '[.INC] e2COMN.INC/LIST'
        INCLUDE '[.INC] FLCOMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
        INCLUDE '[.INC] H2COMN.INC/LIST'
        INCLUDE '[.INC] PRCOMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        DIMENSION YSPEH2(MEQNFL), ENTLH2(MEQNFL), KODEH2(MUMDH2)

C*****************************************************************

C       THIS SUBROUTINE INJECTS THE FUEL AT THE PREVIOUSLY GIVEN
C       LOCATIONS (STORED IN NODEH2)

C*****************************************************************
C
C       IF (KROGER .NE. 1) RETURN
C
C       DETERMINE THE MASS FRACTION OF H2 BASED ON EQUIVALENCE RATIO
        YH2    = PHIEH2/(PHIEH2+34.048)
        KNODH2  = 0

        DO 501 JNODE = 1, NUMDH2
C
C          DETERMINE THE ACTUAL NODE OF INJECTION
C
           INODE  = NODEH2(JNODE)

           IF (CHNGE2(1,INODE) .NE. 0.) THEN
             KNODH2  = KNODH2  + 1
             KODEH2(KNODH2) = INODE
             DO 301 IS = 1, NEQNFL
               DPENG2(IS,INODE) = DPENG2(IS,INODE) + CHNGE2(IS,INODE)
               CHNGE2(IS,INODE) = 0.
301          CONTINUE
           ENDIF
```

```
501     CONTINUE
C
        DO 50 JNODE = 1, KNODH2
C
C         DETERMINE THE ACTUAL NODE OF INJECTION
C
          INODE  = KODEH2(JNODE)
C
C         DETERMINE THE PRIMITIVE VARIABLES BEFORE FUEL INJECTION
C         CORRECTION
C
          RHORPR = DPENG2(1,INODE)
          UCOMPR = DPENG2(2,INODE)/RHORPR
          VCOMPR = DPENG2(3,INODE)/RHORPR
          BEPSPR = DPENG2(4,INODE)
          VELO2U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C         COMPUTE THE DIMENSIONAL QUANTITIES
C
          TEMPPR = TEMPG2(INODE)*TREFFL

C         COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
          SUMY   = 0.
          DO 10 IS = 1, NEQSCH
            JS         = NEQBAS + IS
            YSPEPR(IS) = DPENG2(JS,INODE)/DPENG2(1,INODE)
            IF (YSPEPR(IS) .LT. 0.) THEN
              YSPEPR(IS)       = 0.
              DPENG2(JS,INODE) = 0.
            ENDIF
            SUMY       = SUMY + YSPEPR(IS)
10        CONTINUE

          YSPEPR(NEQSCH+1) = 1. - SUMY
          IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C         DETERMINE THE CURRENT MASS FRACTION OF THE FUEL
          YH2P = YSPEPR(3)
C
C         DETERMINE THE FUEL QUANTITIES
C
          RHOF = RHORPR*(YH2-YH2P)/(1.-YH2)
          PF   = RHOF*RHORFL*UGASFL*RAMWCH(3)*TEMPPR/PRESFL
C
          AMASSX = RHORPR*UCOMPR + RHOF*UCOMPR
          FORCEX = RHORPR*UCOMPR*UCOMPR + RHOF*UCOMPR*UCOMPR +
     1             PRESG2(INODE) + PF

C         DETERMINE THE FACTOR BY WHICH THE MASS FRACTIONS MUST BE
C         ADJUSTED
          YRAT = (1.-YH2)/(1.-YH2P)
C
C         COMPUTE THE NEW MASS FRACTIONS, MOLECULAR MASSES AND THE
C         ENTHALPY OF EACH SPECIES
C
          SYSBMO = 0.
          SYSBMN = 0.
          TENTHI = 0.
```

```
      YSPEH2(3) = YH2
      HM-       = 0.

      DO 20 IS = 1, NSPECH
         IF (IS .NE. 3) YSPEH2(IS) = YSPEPR(IS)*YRAT
         SYSBMO = SYSBMO + YSPEPR(IS)*RAMWCH(IS)
         SYSBMN = SYSBMN + YSPEH2(IS)*RAMWCH(IS)
         ENTLH2(IS) = FMHTCH(IS)*RAMWCH(IS) +
     1            SPCPCH(IS)*(TEMPPR-TREFCH) +
     2        0.5*SPBSCH(IS)*(TEMPPR**2-TREFCH**2)
         TENTHI = TENTHI + ENTLH2(IS)*YSPEPR(IS)
         HM     = HM + FMHTCH(IS)*RAMWCH(IS)*YSPEH2(IS)
20    CONTINUE

      TENTHI = TENTHI/FMREFL + 0.5*VELO2U
      TENTHF = ENTLH2(3)/FMREFL + 0.5*VELO2U
      ENRGYX = UCOMPR*RHORPR*TENTHI + UCOMPR*RHOF*TENTHF

C     MOLECULAR MASSES
      AMASOL = 1./SYSBMO
      AMASNW = 1./SYSBMN

      UGASM  = UGASFL*SYSBMN
      TMRAT  = PRESFL/(RHORFL*UGASM)

C     INITIAL GUESS FOR DENSITY
      RHOM = RHORPR + RHOF
      VM   = VCOMPR

1001  UM   = AMASSX/RHOM
      PM   = FORCEX - UM*AMASSX
      TMD  = PM*TMRAT/RHOM
      VELO2M = 0.5*(UM*UM+VM*VM)
C
C     DETERMINE THE NEW MIXTURE ENTHALPY

      SYSENT = 0.
      DO 30 IS = 1, NSPECH
         ENTLH2(IS) = SPCPCH(IS)*(TMD-TREFCH) +
     2        0.5*SPBSCH(IS)*(TMD**2-TREFCH**2)
         SYSENT = SYSENT + YSPEH2(IS)*ENTLH2(IS)
30    CONTINUE

      ENTHM = (HM+SYSENT)/FMREFL
      RHOMN = ENRGYX/(enthm+VELO2M)/UM
C     COMPA = ABS(RHOM-RHOMN)
      RHOM = RHOMN
C     IF (COMPA .GT. 1.E-4) GOTO 1001

      BENW = (ENTHM+VELO2M)*RHOM - pm
C
C     ADJUST THE DEPENDENT VARIABLES
C
      DPENG2(1,INODE) = RHOM
      DPENG2(2,INODE) = RHOM*UM
      DPENG2(3,INODE) = RHOM*VM
      DPENG2(4,INODE) = BENW
```

```
          DO 40 IS = 1, NEQSCH
            JS          = NEQBAS + IS
            DPENG2(JS,INODE) = RHOM*YSPEH2(IS)
40        CONTINUE

50      CONTINUE
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGCH .NE. 19 .AND. IDBGCH .LT. 1000) RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) RHORPR, RHOM, BEPSPR, BENW,
     1                     AMASOL, AMASNW, UCOMPR, VCOMPR
        DO 60 IS = 1, NSPECH
          WRITE(JDEBUG,1400) YSPEPR(IS), YSPEH2(IS)
60      CONTINUE
        WRITE(JDEBUG,1500) IADDH2, NUMDH2, PHIEH2
        WRITE(JDEBUG,1600)
        WRITE(JDEBUG,1700) (NODEH2(INODE), INODE=1,NUMDH2)
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'-----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM H2MIXT' )
1200    FORMAT(  10X,'-----------------------'/)
1300    FORMAT(15X,'OLD VALUES',6X,'NEW VALUES'/
     1          5X,'DENSITY  ',5X,2G14.6/
     2          5X,'ENERGY   ',5X,2G14.6/
     3          5X,'MOL MASS ',5X,2G14.6/
     4          5X,'VELOCITY ',5X,2G14.6,5X,'SAME U & V COMPONENTS'/)
1400    FORMAT( 5X,'MASS FRAC',5X,2G14.6)
1500    FORMAT(/5X,'IAADH2 =',I5,10X,'NUMDH2 =',I5,
     1         10X,'PHIEH2 =',G14.6)
1600    FORMAT (/10X,'FUEL INJECTION POINTS')
1700    FORMAT (10I7)

        RETURN
        END
```

H2SCRI

```
        SUBROUTINE H2SCRI

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
C       INCLUDE '[.INC] H2COMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'
```

```
      DIMENSION IFNODE(100)
C          -
C*************************************************************************

C     THIS SUBROUTINE OUTPUTS THE DEPENDENT VARIABLES AT A VERTICAL
C     PLANE STARTING FROM A GIVEN NODE AT THE BOTTOM OF THE PLANE.
C     WITH THESE NODES KNOWN, A SCREEN OF FUEL ELEMENTS CAN BE
C     CONSTRUCTED FOR A COMBINATION OF THESE NODES.

C*************************************************************************
C
C     READ THE FOLLOWING FUEL QUANTITIES
C          IBASE   : THE BASE NODE OF THE PLANE OF INJECTION
C
      READ (JREADS,*) IBASE
      INODE = IBASE
      NBTYPE = 0
      NB1    = NEIBG2(4,INODE)
      NB2    = NEIBG2(3,INODE)

      IF (NB1 .NE. 0) THEN
         NBTYPE = 4
         INTYPE = 6
      ELSEIF (NB2 .NE. 0) THEN
         NBTYPE = 3
         INTYPE = 8
      ENDIF
C
C     ERROR CONDITION
C
      IF (NBTYPE .EQ. 0) THEN
         ZER1 = ISTART
         ZER2 = NBTYPE
         CALL ERRORM (46,'H2SCRI','ISTART',ZER1,'NBTYPE',ZER2,JPRINT,
     1            'ERROR IN NEIGHBOUR CELLS OF STARTING POINT')
      ENDIF
C
C     NOW MARCH IN THE APPROPRIATE DIRECTION
C
      KOUNT  = 0

10    KOUNT     = KOUNT + 1
C     SAVE THE NODE WHERE THE FUEL MIGHT BE INJECTED
      IFNODE(KOUNT) = INODE
C     FIND THE NEXT CELL ON TOP OF THE NODE UNDER CONSIDERATION
      NBNEXT        = NEIBG2(NBTYPE,INODE)
C     SEE IF YOU HAVE REACHED THE TOP BOUNDARY SURFACE
      IF (NBNEXT .EQ. 0) GOTO 20
      INODE         = ICELG2(INTYPE,NBNEXT)
      GO TO 10

20    CONTINUE
C     WRITE ALL THE OUTPUT
      OPEN (UNIT=JDUMY1, FILE='H2SCRI.DAT', STATUS='NEW')
      WRITE (JDUMY1,30) KOUNT
30    FORMAT (5X,'TOTAL NODES IN THE PLANE:',I4//
     1         1X,'KOUNT',2X,'NODE',2X,'DENSITY',7X,'U COMP',8X,
```

```
     2              'V COMP',8X,'PRESSURE',4X,'TEMPERATURE')

       DO 40 JNODE = 1, KOUNT
         INODE = IFNODE(JNODE)
         UCOMP = DPENG2(2,INODE)/DPENG2(1,INODE)
         VCOMP = DPENG2(3,INODE)/DPENG2(1,INODE)
         WRITE(JDUMY1,50) JNODE, INODE, DPENG2(1,INODE), UCOMP,
     1                          VCOMP, PRESG2(INODE), TEMPG2(INODE)
40     CONTINUE
50     FORMAT(2I5,5G14.5)

       RETURN
       END
```

H2SCRN

```
       SUBROUTINE H2SCRN

       INCLUDE '[.INC] PRECIS.INC/LIST'
       INCLUDE '[.INC] PARMV2.INC/LIST'
       INCLUDE '[.INC] CHCOMN.INC/LIST'
       INCLUDE '[.INC] FLCOMN.INC/LIST'
       INCLUDE '[.INC] G2COMN.INC/LIST'
C      INCLUDE '[.INC] H2COMN.INC/LIST'
       INCLUDE '[.INC] hexcod.INC     '
       INCLUDE '[.INC] IOCOMN.INC/LIST'
       INCLUDE '[.INC] PRCOMN.INC/LIST'
C
C*********************************************************************

C      THIS SUBROUTINE INITIALIZES THE DEPENDENT VARIABLES FOR FUEL
C      INJECTION AS SCREEN POINTS INSIDE A GIVEN REGION OF AIR FLOW
C      FUEL IS ADDED AS A MIXTURE OF AIR AND AT THE INJECTORS.
C      THE VALUES NEEDED AT THE SCREEN POINTS ARE THE PROPERTIES OF THIS
C      MIXTURE, I.E., TEMPERATURE, PRESSURE, MACH NO., EQUIVALENCE
C      RATIO, AND THE ANGLE OF INJECTION.  ALSO NEEDED IS THE TOTAL
C      NUMBER OF NODES (OR CELLS) AND THE NODES THEMSELVES WHICH ARE
C      THE LOWER CORNERS OF THE INJECTOR CELLS.

C*********************************************************************
C
C      IF (KROGER .NE. 1) RETURN
C
       NBTYPE = 4
       INTYPE = 6
C
C      READ THE FOLLOWING FUEL QUANTITIES
C           TEMPEF  : FUEL TEMPERATURE IN DEGREE K
C           PRESSF  : FUEL PRESSURE IN PASCALS
C           AMACHF  : FUEL MACH NUMBER
C           EQUIVF  : EQUIVALENCE RATIO
C           ANGLEF  : ANGLE OF INJECTION IN DEGREES
C           MNODEF  : NUMBER OF CELLS WITH FUEL INJECTION
```

```
C            INODE   : CELLS WHERE FUEL IS INJECTED
C           -IF EQUIVF > 100 THEN ONLY FUEL IS ADDED AT THE INJECTORS
            -
            READ (JREADS,*) TEMPEF
            READ (JREADS,*) PRESSF
            READ (JREADS,*) AMACHF
            READ (JREADS,*) EQUIVF
            READ (JREADS,*) ANGLEF
            READ (JREADS,*) MNODEF
C
C           COMPUTE THE ANGLE IN RADIANS
            ANGLEF = ANGLEF*3.141592654/180.

C           DETERMINE THE MASS FRACTION OF H2 BASED ON EQUIVALENCE RATIO
C           AND OTHER MASS FRACTIONS
C           YH2 = 2 PHI M_H2 / (M_O2 + 3.76 M_N2 + 2 PHI M_H2)

            YSPEPR(2) = 0.
            YSPEPR(4) = 0.
            IF (EQUIVF .GT. 100.) THEN
               YSPEPR(1)    = 0.
               YSPEPR(3)    = 1.
               YSPEPR(5)    = 0.
            ELSE
               YSPEPR(1)    = 7.93626/(EQUIVF+34.048)
               YSPEPR(3)    = EQUIVF/(EQUIVF+34.048)
               YSPEPR(5)    = 1. - YSPEPR(1) - YSPEPR(3)
            ENDIF
C
C           DETERMINE THE MOLECULAR MASS AND OTHER QUANTITIES FOR THIS MIXTURE
C
            DO 5 IS = 1, NSPECH
                SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
                SYSHFE = SYSHFE + YSPEPR(IS)*FMHTCH(IS)
                SYSCPE = SYSCPE + YSPEPR(IS)*SPCPCH(IS)
                BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
5           CONTINUE

            UGASCO = UGASFL*SYSBMS

C           DETERMINE THE DIMENSIONLESS DENSITY OF THE FUEL MIXTURE
            RHOF   = PRESSF/(UGASCO*TEMPEF*RHORFL)

C           DETERMINE THE DIMENSIONLESS PRESSURE OF THE FUEL MIXTURE
            PRESSF = PRESSF/PRESFL
C
C           DETERMINE GAMMA FOR THIS MIXTURE
            BIGAMT = BIGAM*TEMPEF
            SYSCVE = SYSCPE + BIGAMT - UGASFL*SYSBMS
            GAMMAE = (SYSCPE + BIGAMT)/SYSCVE

C           DETERMINE THE OVERALL DIMENSIONLESS VELOCITY OF THE FUEL
            VELOF  = AMACHF*SQRT(GAMMAE*PRESSF/RHOF)
            UCOMPF = VELOF*COS(ANGLEF)
            VCOMPF = VELOF*SIN(ANGLEF)
            VELO2I = UCOMPF*UCOMPF + VCOMPF*VCOMPF
```

```
C       DETERMINE THE ENERGY TERM
        BEE    = SYSHFE + (TEMPEF-TREFCH)*SYSCPE - UGASFL*TEMPEF*SYSBMS
     1       -                  + 0.5*(TEMPEF*TEMPEF-TREFCH*TREFCH)*BIGAM
        BEE    = BEE/FMREFL + 0.5*VEL02I
C
        TEMPEF = TEMPEF/TREFFL
C
C
C                        -------------
C                        |           |
C                        |  IUCELL   |
C                        |           |
C                        ------------- I2NODE
C                        |           |
C                        |  IOCELL   |
C                        |           |
C                        ------------- I1NODE
C                        |           |
C                        |  ILCELL   |
C                        |           |
C                        -------------
C
C
        NNODEF = ABS(MNODEF)
        DO 50 JCELL = 1, NNODEF
C          READ THE LOWER NODE OF THIS CELL
           READ (JREADS,*) I1NODE
           IOCELL = NEIBG2(4,I1NODE)
           IF (IOCELL .EQ. 0) GOTO 50
           ILCELL = NEIBG2(1,I1NODE)
           I2NODE = ICELG2(6,IOCELL)
           IUCELL = NEIBG2(4,I1NODE)
           KAUXG2(IOCELL) = IOR(KAUXG2(IOCELL),KL1000)
C          SET THE DEPENDENT VARIABLES
           DPENG2(1,I1NODE) = RHOF
           DPENG2(1,I2NODE) = RHOF
           DPENG2(2,I1NODE) = RHOF*UCOMPF
           DPENG2(2,I2NODE) = RHOF*UCOMPF
           DPENG2(3,I1NODE) = RHOF*VCOMPF
           DPENG2(3,I2NODE) = RHOF*VCOMPF
           DPENG2(4,I1NODE) = BEE*RHOF
           DPENG2(4,I2NODE) = BEE*RHOF

           IF (MNODEF .LT. 0) THEN
             ICRITE = NEIBG2(3,I1NODE)
             DPENG2(1,ICELG2(4,ICRITE)) = RHOF
             DPENG2(2,ICELG2(4,ICRITE)) = RHOF*UCOMPF
             DPENG2(3,ICELG2(4,ICRITE)) = RHOF*VCOMPF
             DPENG2(4,ICELG2(4,ICRITE)) = BEE*RHOF
             DPENG2(1,ICELG2(6,ICRITE)) = RHOF
             DPENG2(2,ICELG2(6,ICRITE)) = RHOF*UCOMPF
             DPENG2(3,ICELG2(6,ICRITE)) = RHOF*VCOMPF
             DPENG2(4,ICELG2(6,ICRITE)) = BEE*RHOF
           ENDIF

           DO 10 JS = NEQBAS+1, NEQNFL
              IS               = JS - NEQBAS
```

```
            DPENG2(JS,I1NODE) = RHOF*YSPEPR(IS)
            DPENG2(JS,I2NODE) = RHOF*YSPEPR(IS)
            IF (MNODEF .LT. 0) THEN
               DPENG2(JS,ICELG2(4,ICRITE)) = RHOF*YSPEPR(IS)
               DPENG2(JS,ICELG2(6,ICRITE)) = RHOF*YSPEPR(IS)
            ENDIF
10          CONTINUE

            PRESG2(I1NODE) = PRESSF
            TEMPG2(I1NODE) = TEMPEF
            PRESG2(I2NODE) = PRESSF
            TEMPG2(I2NODE) = TEMPEF

C           SET THE BOUNDARY CONDITION POINTER

            DO 20 JS = 1, NBNDG2
               IF (IBNDG2(1,JS) .EQ. I1NODE) THEN
                  IBNDG2(5,JS) = 2
                  GOTO 30
               ENDIF
20          CONTINUE

            NBNDG2          = NBNDG2 + 1
            IBNDG2(1,NBNDG2) = I1NODE
            IBNDG2(2,NBNDG2) = IOCELL
            IBNDG2(3,NBNDG2) = ILCELL
            IBNDG2(4,NBNDG2) = 0
            IBNDG2(5,NBNDG2) = 2

30          DO 40 JS = 1, NBNDG2
               IF (IBNDG2(1,JS) .EQ. I2NODE) THEN
                  IBNDG2(5,JS) = 2
                  GOTO 50
               ENDIF
40          CONTINUE

            NBNDG2          = NBNDG2 + 1
            IBNDG2(1,NBNDG2) = I2NODE
            IBNDG2(2,NBNDG2) = IUCELL
            IBNDG2(3,NBNDG2) = IOCELL
            IBNDG2(4,NBNDG2) = 0
            IBNDG2(5,NBNDG2) = 2

50       CONTINUE
C

C        FOR ROGERS AND CHINITZ MODEL THE NUMBER OF EQUATIONS MUST
C        BE ADJUSTED

         IF (KROGER .EQ. 1 .AND. NINRCH .GT. 0) THEN
            DO 200 INODE = 1, NNODG2

C              COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
               SUMY   = 0.
               YUPPER = 1. - YNRTCH
               RHORPR = DPENG2(1,INODE)
```

```
            DO 190 IS = 1, NEQSCH
               JS         = NEQBAS + IS
               YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
               IF (YSPEPR(IS) .LT. 0.) THEN
                 YSPEPR(IS)       = 0.
                 DPENG2(JS,INODE) = 0.
               ENDIF
c              IF (YSPEPR(IS) .GT. YUPPER) THEN
c                YSPEPR(IS)       = YUPPER
c                DPENG2(JS,INODE) = YUPPER*RHORPR
c              ENDIF
               SUMY       = SUMY + YSPEPR(IS)
190         CONTINUE
C           THE FOLLOWING IS FOR SPECIES 4 = NEQSCH+1
            YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
            IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C           ADJUST THE NEWLY DEFINED VARIABLE AT THIS NODE
            DPENG2(NEQNFL+1,INODE) = RHORPR*YSPEPR(NEQSCH+1)
200       CONTINUE
C         NOW ADJUST THE NUMBER OF EQUATIONS
          YNRTCH = 0.
          NEQNFL = NEQNFL + 1
          NEQSCH = NEQSCH + 1
          NINRCH = NINRCH - 1
        ENDIF
C
        RETURN
        END
```

H3SCRN

```
        SUBROUTINE H2SCRN

        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] CHCOMN.INC/LIST'
        INCLUDE '[.INC] FLCOMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
C       INCLUDE '[.INC] H2COMN.INC/LIST'
        INCLUDE '[.INC] hexcod.INC     '
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        INCLUDE '[.INC] PRCOMN.INC/LIST'
C
C*********************************************************************

C       THIS SUBROUTINE INITIALIZES THE DEPENDENT VARIABLES FOR FUEL
C       INJECTION AS SCREEN POINTS INSIDE A GIVEN REGION OF AIR FLOW
C       FUEL IS ADDED AS A MIXTURE OF AIR AND AT THE INJECTORS.
C       THE VALUES NEEDED AT THE SCREEN POINTS ARE THE PROPERTIES OF THIS
C       MIXTURE, I.E., TEMPERATURE, PRESSURE, MACH NO., EQUIVALENCE
C       RATIO, AND THE ANGLE OF INJECTION.  ALSO NEEDED IS THE TOTAL
C       NUMBER OF NODES (OR CELLS) AND THE NODES THEMSELVES WHICH ARE
C       THE LOWER CORNERS OF THE INJECTOR CELLS.
```

```
C*********************************************************************
C
C       IF (KROGER .NE. 1) RETURN
C
        NBTYPE = 4
        INTYPE = 6
C
C       READ THE FOLLOWING FUEL QUANTITIES
C            TEMPEF  : FUEL TEMPERATURE IN DEGREE K
C            PRESSF  : FUEL PRESSURE IN PASCALS
C            AMACHF  : FUEL MACH NUMBER
C            EQUIVF  : EQUIVALENCE RATIO
C            ANGLEF  : ANGLE OF INJECTION IN DEGREES
C            MNODEF  : NUMBER OF CELLS WITH FUEL INJECTION
C            INODE   : CELLS WHERE FUEL IS INJECTED
C            IF EQUIVF > 100 THEN ONLY FUEL IS ADDED AT THE INJECTORS

        READ (JREADS,*) TEMPEF
        READ (JREADS,*) PRESSF
        READ (JREADS,*) AMACHF
        READ (JREADS,*) EQUIVF
        READ (JREADS,*) ANGLEF
        READ (JREADS,*) MNODEF
C
C       COMPUTE THE ANGLE IN RADIANS
        ANGLEF = ANGLEF*3.141592654/180.

C       DETERMINE THE MASS FRACTION OF H2 BASED ON EQUIVALENCE RATIO
C       AND OTHER MASS FRACTIONS
C       YH2 = 2 PHI M_H2 / (M_O2 + 3.76 M_N2 + 2 PHI M_H2)

        YSPEPR(2) = 0.
        YSPEPR(4) = 0.
        IF (EQUIVF .GT. 100.) THEN
           YSPEPR(1)    = 0.
           YSPEPR(3)    = 1.
           YSPEPR(5)    = 0.
        ELSE
           YSPEPR(1)    = 7.93626/(EQUIVF+34.048)
           YSPEPR(3)    = EQUIVF/(EQUIVF+34.048)
           YSPEPR(5)    = 1. - YSPEPR(1) - YSPEPR(3)
        ENDIF
C
C       DETERMINE THE MOLECULAR MASS AND OTHER QUANTITIES FOR THIS MIXTURE
C
        DO 5 IS = 1, NSPECH
            SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
            SYSHFE = SYSHFE + YSPEPR(IS)*FMHTCH(IS)
            SYSCPE = SYSCPE + YSPEPR(IS)*SPCPCH(IS)
            BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
5       CONTINUE

        UGASCO = UGASFL*SYSBMS

C       DETERMINE THE DIMENSIONLESS DENSITY OF THE FUEL MIXTURE
        RHOF   = PRESSF/(UGASCO*TEMPEF*RHORFL)
```

```
C       DETERMINE THE DIMENSIONLESS PRESSURE OF THE FUEL MIXTURE
        PRESSF = PRESSF/PRESFL
C
C       DETERMINE GAMMA FOR THIS MIXTURE
        BIGAMT = BIGAM*TEMPEF
        SYSCVE = SYSCPE + BIGAMT - UGASFL*SYSBMS
        GAMMAE = (SYSCPE + BIGAMT)/SYSCVE

C       DETERMINE THE OVERALL DIMENSIONLESS VELOCITY OF THE FUEL
        VELOF  = AMACHF*SQRT(GAMMAE*PRESSF/RHOF)
        UCOMPF = VELOF*COS(ANGLEF)
        VCOMPF = VELOF*SIN(ANGLEF)
        VEL02I = UCOMPF*UCOMPF + VCOMPF*VCOMPF

C       DETERMINE THE ENERGY TERM
        BEE    = SYSHFE + (TEMPEF-TREFCH)*SYSCPE - UGASFL*TEMPEF*SYSBMS
     1                        + 0.5*(TEMPEF*TEMPEF-TREFCH*TREFCH)*BIGAM
        BEE    = BEE/FMREFL + 0.5*VEL02I
C
        TEMPEF = TEMPEF/TREFFL
C
C
C                        -------------
C                        |           |
C                        |  IUCELL   |
C                        |           |
C                        ------------- KNE
C                        |           |
C                        |  IOCELL   |
C                        |           |
C                        ------------- KSE
C                        |           |
C                        |  ILCELL   |
C                        |           |
C                        -------------
C
C
        NNODEF = ABS(MNODEF)
        DO 55 JCELL = 1, NNODEF
C          READ THE LOWER NODE OF THIS CELL
           READ (JREADS,*) KSE
           IOCELL = NEIBG2(4,KSE)
           IF (IOCELL .EQ. 0) GOTO 55
           ILCELL = NEIBG2(1,KSE)
           KNE    = ICELG2(6,IOCELL)
           IUCELL = NEIBG2(4,KNE)
           KNW    = ICELG2(8,IOCELL)
           KSW    = ICELG2(2,IOCELL)

           KAUXG2(IOCELL) = IOR(KAUXG2(IOCELL),KL1000)
C          SET THE DEPENDENT VARIABLES
           DPENG2(1,KSE) = RHOF
           DPENG2(1,KNE) = RHOF
           DPENG2(2,KSE) = RHOF*UCOMPF
           DPENG2(2,KNE) = RHOF*UCOMPF
           DPENG2(3,KSE) = RHOF*VCOMPF
```

```
      DPENG2(3,KNE) = RHOF*VCOMPF
      DPENG2(4,KSE) = BEE*RHOF
      DPENG2(4,KNE) = BEE*RHOF
      DPENG2(2,KSW) = 0.
      DPENG2(3,KSW) = 0.
      DPENG2(2,KNW) = 0.
      DPENG2(3,KNW) = 0.

      IF (MNODEF .LT. 0) THEN
        ICRITE = NEIBG2(3,KSE)
        DPENG2(1,ICELG2(4,ICRITE)) = RHOF
        DPENG2(2,ICELG2(4,ICRITE)) = RHOF*UCOMPF
        DPENG2(3,ICELG2(4,ICRITE)) = RHOF*VCOMPF
        DPENG2(4,ICELG2(4,ICRITE)) = BEE*RHOF
        DPENG2(1,ICELG2(6,ICRITE)) = RHOF
        DPENG2(2,ICELG2(6,ICRITE)) = RHOF*UCOMPF
        DPENG2(3,ICELG2(6,ICRITE)) = RHOF*VCOMPF
        DPENG2(4,ICELG2(6,ICRITE)) = BEE*RHOF
      ENDIF

      DO 10 JS = NEQBAS+1, NEQNFL
         IS                = JS - NEQBAS
         DPENG2(JS,KSE) = RHOF*YSPEPR(IS)
         DPENG2(JS,KNE) = RHOF*YSPEPR(IS)
         IF (MNODEF .LT. 0) THEN
            DPENG2(JS,ICELG2(4,ICRITE)) = RHOF*YSPEPR(IS)
            DPENG2(JS,ICELG2(6,ICRITE)) = RHOF*YSPEPR(IS)
         ENDIF
10    CONTINUE

      PRESG2(KSE) = PRESSF
      TEMPG2(KSE) = TEMPEF
      PRESG2(KNE) = PRESSF
      TEMPG2(KNE) = TEMPEF

C     SET THE BOUNDARY CONDITION POINTER

      DO 20 JS = 1, NBNDG2
         IF (IBNDG2(1,JS) .EQ. KSE) THEN
            IBNDG2(5,JS) = 2
            GOTO 30
         ENDIF
20    CONTINUE

      NBNDG2           = NBNDG2 + 1
      IBNDG2(1,NBNDG2) = KSE
      IBNDG2(2,NBNDG2) = IOCELL
      IBNDG2(3,NBNDG2) = ILCELL
      IBNDG2(4,NBNDG2) = 0
      IBNDG2(5,NBNDG2) = 2

30    DO 40 JS = 1, NBNDG2
         IF (IBNDG2(1,JS) .EQ. KNE) THEN
            IBNDG2(5,JS) = 2
            GOTO 50
         ENDIF
40    CONTINUE
```

```
            NBNDG2           = NBNDG2 + 1
            IBNDG2(1,NBNDG2) = KNE
            IBNDG2(2,NBNDG2) = IUCELL
            IBNDG2(3,NBNDG2) = IOCELL
            IBNDG2(4,NBNDG2) = 0
            IBNDG2(5,NBNDG2) = 2
50          CONTINUE

            DO 51 JS = 1, NBNDG2
               IF (IBNDG2(1,JS) .EQ. KNW) THEN
                  IBNDG2(5,JS) = 11
                  GOTO 52
               ENDIF
51          CONTINUE

            NBNDG2           = NBNDG2 + 1
            IBNDG2(1,NBNDG2) = KNW
            IBNDG2(2,NBNDG2) = IUCELL
            IBNDG2(3,NBNDG2) = IOCELL
            IBNDG2(4,NBNDG2) = 0
            IBNDG2(5,NBNDG2) = 11

52          DO 53 JS = 1, NBNDG2
               IF (IBNDG2(1,JS) .EQ. KSW) THEN
                  IBNDG2(5,JS) = 11
                  GOTO 55
               ENDIF
53          CONTINUE

            NBNDG2           = NBNDG2 + 1
            IBNDG2(1,NBNDG2) = KSW
            IBNDG2(2,NBNDG2) = IOCELL
            IBNDG2(3,NBNDG2) = ILCELL
            IBNDG2(4,NBNDG2) = 0
            IBNDG2(5,NBNDG2) = 11

55       CONTINUE
C
C        FOR ROGERS AND CHINITZ MODEL THE NUMBER OF EQUATIONS MUST
C        BE ADJUSTED

         IF (KROGER .EQ. 1 .AND. NINRCH .GT. 0) THEN
            DO 200 INODE = 1, NNODG2

C              COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
               SUMY   = 0.
               YUPPER = 1. - YNRTCH
               RHORPR = DPENG2(1,INODE)

               DO 190 IS = 1, NEQSCH
                  JS         = NEQBAS + IS
                  YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
                  IF (YSPEPR(IS) .LT. 0.) THEN
                    YSPEPR(IS)       = 0.
                    DPENG2(JS,INODE) = 0.
                  ENDIF
```

```
c               IF (YSPEPR(IS) .GT. YUPPER) THEN
c                 YSPEPR(IS)       = YUPPER
c                 DPENG2(JS,INODE) = YUPPER*RHORPR
c               ENDIF
                SUMY       = SUMY + YSPEPR(IS)
190           CONTINUE
C             THE FOLLOWING IS FOR SPECIES 4 = NEQSCH+1
              YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
              IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C             ADJUST THE NEWLY DEFINED VARIABLE AT THIS NODE
              DPENG2(NEQNFL+1,INODE) = RHORPR*YSPEPR(NEQSCH+1)
200           CONTINUE
C           NOW ADJUST THE NUMBER OF EQUATIONS
            YNRTCH = 0.
            NEQNFL = NEQNFL + 1
            NEQSCH = NEQSCH + 1
            NINRCH = NINRCH - 1
          ENDIF
C
          CALL A2CEWC
          RETURN
          END
```

H2SOLF

```
          SUBROUTINE H2SOLF (ITGL)

          INCLUDE '[.INC] PRECIS.INC/LIST'
          INCLUDE '[.INC] PARMV2.INC/LIST'
          INCLUDE '[.INC] CHCOMN.INC/LIST'
          INCLUDE '[.INC] e2COMN.INC/LIST'
          INCLUDE '[.INC] FLCOMN.INC/LIST'
          INCLUDE '[.INC] G2COMN.INC/LIST'
          INCLUDE '[.INC] H2COMN.INC/LIST'
          INCLUDE '[.INC] M2COMN.INC/LIST'
          INCLUDE '[.INC] IOCOMN.INC/LIST'
          INCLUDE '[.INC] TICOMN.INC/LIST'
          DIMENSION YSPEH2(MEQNFL), ENTLH2(MEQNFL), KODEH2(MUMDH2)

          DIMENSION BIGFS (MEQNFL)        ,   BIGFE (MEQNFL)        ,
     1            BIGFN (MEQNFL)        ,   BIGFW (MEQNFL)        ,
     2            BIGGS (MEQNFL)        ,   BIGGE (MEQNFL)        ,
     3            BIGGN (MEQNFL)        ,   BIGGW (MEQNFL)        ,
     4            DPENFA(MEQNFL,4)

C*********************************************************************

C         THIS SUBROUTINE STEPS THROUGH EACH FUEL CELL ON THE TEMPORAL
C         LEVEL ITGL AND APPLIES NI'S SCHEME, I.E., INTEGRATES OVER ALL
C         THE FUEL CELLS ON LEVEL ITGL.  THIS FOR EMBEDDED FUEL INLET
C         MODELLING.

C*********************************************************************
```

```
C
      YH2 = PHIEH2/(PHIEH2+34.048)

      DO 50 JCELL = 1, NCELH2
C
C        DETERMINE THE ACTUAL NODE OF INJECTION
C
         ICELL  = ICELH2(JCELL)

         IF (ICELL.LT.ILVLTI(1,ITGL) .OR. ICELL.GT.ILVLTI(1,ITGL))
     1            GOTO 50

C        SET UP NODE POINTERS FOR THIS CELL

         KSW = ICELG2( 2,ICELL)
         KSE = ICELG2( 4,ICELL)
         KNE = ICELG2( 6,ICELL)
         KNW = ICELG2( 8,ICELL)

C        --------
C        GEOMETRY
C        --------
C
C        GEOMETRY OF ALL CELL CORNERS

         XSW = GEOMG2(1,KSW)
         YSW = GEOMG2(2,KSW)

         XSE = GEOMG2(1,KSE)
         YSE = GEOMG2(2,KSE)

         XNE = GEOMG2(1,KNE)
         YNE = GEOMG2(2,KNE)

         XNW = GEOMG2(1,KNW)
         YNW = GEOMG2(2,KNW)
C
C        THE RATIO DELTA-t TO CELL VOLUME
         DTDVOL = CELLTI(ICELL)*RVOLM2(ICELL)

         RHORSW = DPENG2(1,KSW)
         RHORNW = DPENG2(1,KNW)
         RHORNE = DPENG2(1,KNE)
         RHORSE = DPENG2(1,KSE)

         BEPSSW = DPENG2(4,KSW)
         BEPSNW = DPENG2(4,KNW)
         BEPSNE = DPENG2(4,KNE)
         BEPSSE = DPENG2(4,KSE)

         UCOMSW = DPENG2(2,KSW)/RHORSW
         VCOMSW = DPENG2(3,KSW)/RHORSW
         UCOMNW = DPENG2(2,KNW)/RHORNW
         VCOMNW = DPENG2(3,KNW)/RHORNW
         UCOMNE = DPENG2(2,KNE)/RHORNE
         VCOMNE = DPENG2(3,KNE)/RHORNE
         UCOMSE = DPENG2(2,KSE)/RHORSE
```

```
      VCOMSE = DPENG2(3,KSE)/RHORSE

      PRESSW = PRESG2(KSW)
      PRESSE = PRESG2(KSE)
      PRESNE = PRESG2(KNE)
      PRESNW = PRESG2(KNW)

      TEMPSE = TEMPG2(KSE)*TREFFL
      TEMPNE = TEMPG2(KSE)*TREFFL

C     DETERMINE THE FUEL QUANTITIES
C
      YH2PSE = DPENG2(7,KSE)/RHORSE
      YH2PNE = DPENG2(7,KNE)/RHORNE

      RHOFSE = RHORSE*(YH2-YH2PSE)/(1.-YH2)
      RHOFNE = RHORNE*(YH2-YH2PNE)/(1.-YH2)
      PFSE   = RHOFSE*RHORFL*UGASFL*RAMWCH(3)*TEMPSE/PRESFL
      PFNE   = RHOFNE*RHORFL*UGASFL*RAMWCH(3)*TEMPNE/PRESFL
C
      RHORSE = RHORSE - RHOFSE
      RHORNE = RHORNE - RHOFNE

      PRESSE = PRESSE - PFSE
      PRESNE = PRESNE - PFNE

      BEPSSE = (BEPSSW + PRESSW)*UCOMSW/UCOMSE - PRESSE
      BEPSNE = (BEPSNW + PRESNW)*UCOMNW/UCOMNE - PRESNE
C
C
C     COMPUTE THE DEPENDENT VARIABLES AT THE FACES

      PRESSS = 0.5*( PRESSW + PRESSE )
      PRESSE = 0.5*( PRESSE + PRESNE )
      PRESSN = 0.5*( PRESNW + PRESNE )
      PRESSW = 0.5*( PRESSW + PRESNW )

      DPENFA(1,1) = 0.5*( RHORSW + RHORSE )
      DPENFA(1,2) = 0.5*( RHORSE + RHORNE )
      DPENFA(1,3) = 0.5*( RHORNE + RHORNW )
      DPENFA(1,4) = 0.5*( RHORNW + RHORSW )

      DPENFA(2,1) = 0.5*( RHORSW*UCOMSW + RHORSE*UCOMSE )
      DPENFA(2,2) = 0.5*( RHORSE*UCOMSE + RHORNE*UCOMNE )
      DPENFA(2,3) = 0.5*( RHORNE*UCOMNE + RHORNW*UCOMNW )
      DPENFA(2,4) = 0.5*( RHORNW*UCOMNW + RHORSW*UCOMSW )

      DPENFA(3,1) = 0.5*( RHORSW*VCOMSW + RHORSE*VCOMSE )
      DPENFA(3,2) = 0.5*( RHORSE*VCOMSE + RHORNE*VCOMNE )
      DPENFA(3,3) = 0.5*( RHORNE*VCOMNE + RHORNW*VCOMNW )
      DPENFA(3,4) = 0.5*( RHORNW*VCOMNW + RHORSW*VCOMSW )

      DPENFA(4,1) = 0.5*( BEPSSW + BEPSSE )
      DPENFA(4,2) = 0.5*( BEPSSE + BEPSNE )
      DPENFA(4,3) = 0.5*( BEPSNE + BEPSNW )
      DPENFA(4,4) = 0.5*( BEPSNW + BEPSSW )
```

```
      UCOMPS = DPENFA(2,1)/DPENFA(1,1)
      VCOMPS = DPENFA(3,1)/DPENFA(1,1)
      UCOMPE = DPENFA(2,2)/DPENFA(1,2)
      VCOMPE = DPENFA(3,2)/DPENFA(1,2)
      UCOMPN = DPENFA(2,3)/DPENFA(1,3)
      VCOMPN = DPENFA(3,3)/DPENFA(1,3)
      UCOMPW = DPENFA(2,4)/DPENFA(1,4)
      VCOMPW = DPENFA(3,4)/DPENFA(1,4)

C     ----------
C     FLUX TERMS
C     ----------
C     SOUTH

      BIGFS(1) = DPENFA(2,1)
      BIGFS(2) = DPENFA(2,1)*UCOMPS + PRESSS
      BIGFS(3) = DPENFA(2,1)*VCOMPS
      BIGFS(4) = UCOMPS*(DPENFA(4,1) + PRESSS)

      BIGGS(1) = DPENFA(3,1)
      BIGGS(2) = BIGFS(3)
      BIGGS(3) = DPENFA(3,1)*VCOMPS + PRESSS
      BIGGS(4) = VCOMPS*(DPENFA(4,1) + PRESSS)

C     EAST

      BIGFE(1) = DPENFA(2,2)
      BIGFE(2) = DPENFA(2,2)*UCOMPE + PRESSE
      BIGFE(3) = DPENFA(2,2)*VCOMPE
      BIGFE(4) = UCOMPE*(DPENFA(4,2) + PRESSE)
      BIGGE(1) = DPENFA(3,2)
      BIGGE(2) = BIGFE(3)
      BIGGE(3) = DPENFA(3,2)*VCOMPE + PRESSE
      BIGGE(4) = VCOMPE*(DPENFA(4,2) + PRESSE)

C     NORTH

      BIGFN(1) = DPENFA(2,3)
      BIGFN(2) = DPENFA(2,3)*UCOMPN + PRESSN
      BIGFN(3) = DPENFA(2,3)*VCOMPN
      BIGFN(4) = UCOMPN*(DPENFA(4,3) + PRESSN)
      BIGGN(1) = DPENFA(3,3)
      BIGGN(2) = BIGFN(3)
      BIGGN(3) = DPENFA(3,3)*VCOMPN + PRESSN
      BIGGN(4) = VCOMPN*(DPENFA(4,3) + PRESSN)

C     WEST

      BIGFW(1) = DPENFA(2,4)
      BIGFW(2) = DPENFA(2,4)*UCOMPW + PRESSW
      BIGFW(3) = DPENFA(2,4)*VCOMPW
      BIGFW(4) = UCOMPW*(DPENFA(4,4) + PRESSW)
      BIGGW(1) = DPENFA(3,4)
      BIGGW(2) = BIGFW(3)
      BIGGW(3) = DPENFA(3,4)*VCOMPW + PRESSW
      BIGGW(4) = VCOMPW*(DPENFA(4,4) + PRESSW)
```

```
C          ----------------------------
C          FIRST ORDER CELL CHANGE DUCELL
C          ----------------------------

C          CALCULATE CHANGE AT CELL CENTER BY PERFORMING A FLUX BALANCE
C          AND DO DISTRIBUTION

           DO 120 J = 1, 4
              DUCELL = 0.25*DTDVOL*(
     1                 BIGFW(J)*(YNW-YSW) - BIGGW(J)*(XNW-XSW) +
     1                 BIGFN(J)*(YNE-YNW) - BIGGN(J)*(XNE-XNW) +
     1                 BIGFE(J)*(YSE-YNE) - BIGGE(J)*(XSE-XNE) +
     1                 BIGFS(J)*(YSW-YSE) - BIGGS(J)*(XSW-XSE) )

              CHNGE2(J,KSW) = CHNGE2(J,KSW) + DUCELL
              CHNGE2(J,KNW) = CHNGE2(J,KNW) + DUCELL

120        CONTINUE

C          CALCULATE CHANGES AT WESTERN NODES FOR SPECIES EQUATIONS

           DO 130 J = NEQBAS+1, NEQNFL
             CHNGE2(J,KSW) = CHNGE2(1,KSW)*DPENG2(J,KSW)/DPENG2(1,KSW)
             CHNGE2(J,KNW) = CHNGE2(1,KNW)*DPENG2(J,KNW)/DPENG2(1,KNW)
130        CONTINUE

50      CONTINUE
C
        RETURN
        END
```

H2TRIN

```
        SUBROUTINE H2TRIN
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'CHCOMN.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'FLCOMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'PRCOMN.INC'
C
C**********************************************************************

C       THIS SUBROUTINE INITIALIZES THE DEPENDENT VARIABLES FOR FUEL
C       INJECTION AS WALL POINTS FOR INTERNAL BOUNDARIES FOR A MIXTURE OF
C       FUEL AND AIR.  THE VALUES NEEDED AT THE INTERNAL POINTS ARE THE
C       PROPERTIES OF THIS MIXTURE, I.E., TEMPERATURE, PRESSURE, MACH NO.,
C       EQUIVALENCE RATIO, AND THE ANGLE OF INJECTION.  ALSO NEEDED IS THE
C       TOTAL NUMBER OF INJECTION POINTS AND THE ACTUAL NODE NUMBERS.
```

```
C***********************************************************************
C
C       IF (KROGER .NE. 1) RETURN
C
C       READ THE FOLLOWING FUEL QUANTITIES
C            TEMPEF  : FUEL TEMPERATURE IN DEGREE K
C            PRESSF  : FUEL PRESSURE IN PASCALS
C            AMACHF  : FUEL MACH NUMBER
C            EQUIVF  : EQUIVALENCE RATIO
C            ANGLEF  : ANGLE OF INJECTION IN DEGREES
C            NINJEC  : NUMBER OF CELLS WITH FUEL INJECTION
C            INODE   : CELLS WHERE FUEL IS INJECTED
C            IF EQUIVF > 100 THEN ONLY FUEL IS ADDED AT THE INJECTORS
C
        READ (JREADS,*) TEMPEF
        READ (JREADS,*) PRESSF
        READ (JREADS,*) AMACHF
        READ (JREADS,*) EQUIVF
        READ (JREADS,*) ANGLEF
        READ (JREADS,*) NINJEC
C
C       COMPUTE THE ANGLE IN RADIANS
        ANGLEF = ANGLEF*3.141592654/180.

C       DETERMINE THE MASS FRACTION OF H2 BASED ON EQUIVALENCE RATIO
C       AND OTHER MASS FRACTIONS
C       YH2 = 2 PHI M_H2 / (M_O2 + 3.76 M_N2 + 2 PHI M_H2)

        YSPEPR(2) = 0.
        YSPEPR(4) = 0.
        IF (EQUIVF .GT. 100.) THEN
           YSPEPR(1)    = 0.
           YSPEPR(3)    = 1.
           YSPEPR(5)    = 0.
        ELSE
           YSPEPR(1)    = 7.93626/(EQUIVF+34.048)
           YSPEPR(3)    = EQUIVF/(EQUIVF+34.048)
           YSPEPR(5)    = 1. - YSPEPR(1) - YSPEPR(3)
        ENDIF
C
C       DETERMINE THE MOLECULAR MASS AND OTHER QUANTITIES FOR THIS MIXTURE
C
        SYSBMS = 0.
        SYSHFE = 0.
        SYSCPE = 0.
        BIGAM  = 0.
        DO 5 IS = 1, NSPECH
            SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
            SYSHFE = SYSHFE + YSPEPR(IS)*FMHTCH(IS)
            SYSCPE = SYSCPE + YSPEPR(IS)*SPCPCH(IS)
            BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
5       CONTINUE

        UGASCO = UGASFL*SYSBMS

C       DETERMINE THE DIMENSIONLESS DENSITY OF THE FUEL MIXTURE
        RHOF   = PRESSF/(UGASCO*TEMPEF*RHORFL)
```

```
C       DETERMINE THE DIMENSIONLESS PRESSURE OF THE FUEL MIXTURE
        PRESSF = PRESSF/PRESFL
C
C       DETERMINE GAMMA FOR THIS MIXTURE
        BIGAMT = BIGAM*TEMPEF
        SYSCVE = SYSCPE + BIGAMT - UGASFL*SYSBMS
        GAMMAE = (SYSCPE + BIGAMT)/SYSCVE

C       DETERMINE THE OVERALL DIMENSIONLESS VELOCITY OF THE FUEL
        VELOF  = AMACHF*SQRT(GAMMAE*PRESSF/RHOF)
        UCOMPF = VELOF*COS(ANGLEF)
        VCOMPF = VELOF*SIN(ANGLEF)
        VELO2I = UCOMPF*UCOMPF + VCOMPF*VCOMPF

C       DETERMINE THE ENERGY TERM
        BEE    = SYSHFE + (TEMPEF-TREFCH)*SYSCPE - UGASFL*TEMPEF*SYSBMS
     1                  + 0.5*(TEMPEF*TEMPEF-TREFCH*TREFCH)*BIGAM
        BEE    = BEE/FMREFL + 0.5*VELO2I
C
        TEMPEF = TEMPEF/TREFFL

        DO 30 JCELL = 1, NINJEC
C          READ THE NODE NUMBER FOR THIS VALUE
           READ (JREADS,*) INODEF

C          SET THE DEPENDENT VARIABLES AT THIS NODE
           DPENG2(1,INODEF) = RHOF
           DPENG2(2,INODEF) = RHOF*UCOMPF
           DPENG2(3,INODEF) = RHOF*VCOMPF
           DPENG2(4,INODEF) = BEE*RHOF

           DO 10 JS = NEQBAS+1, NEQNFL
              IS             = JS - NEQBAS
              DPENG2(JS,INODEF) = RHOF*YSPEPR(IS)
10         CONTINUE

           PRESG2(INODEF) = PRESSF
           TEMPG2(INODEF) = TEMPEF

C          SET THE BOUNDARY CONDITION POINTER

           DO 20 JS = 1, NBNDG2
              IF (IBNDG2(1,JS) .EQ. INODEF) THEN
                 IBNDG2(5,JS) = 2
                 GOTO 30
              ENDIF
20         CONTINUE

           NBNDG2           = NBNDG2 + 1
           IBNDG2(1,NBNDG2) = INODEF
           IBNDG2(2,NBNDG2) = 0
           IBNDG2(3,NBNDG2) = 0
           IBNDG2(4,NBNDG2) = 0
           IBNDG2(5,NBNDG2) = 2
30      CONTINUE
C
```

```
C       FOR ROGERS AND CHINITZ MODEL THE NUMBER OF EQUATIONS MUST
C       BE ADJUSTED

        IF (KROGER .EQ. 1 .AND. NINRCH .GT. 0) THEN
           DO 200 INODE = 1, NNODG2

C             COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
              SUMY   = 0.
              YUPPER = 1. - YNRTCH
              RHORPR = DPENG2(1,INODE)

              DO 190 IS = 1, NEQSCH
                 JS         = NEQBAS + IS
                 YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
                 IF (YSPEPR(IS) .LT. 0.) THEN
                   YSPEPR(IS)       = 0.
                   DPENG2(JS,INODE) = 0.
                 ENDIF
                 IF (YSPEPR(IS) .GT. 1.) THEN
                   YSPEPR(IS)       = 1.
                   DPENG2(JS,INODE) = YUPPER*RHORPR
                 ENDIF
                 SUMY       = SUMY + YSPEPR(IS)
190           CONTINUE
C             THE FOLLOWING IS FOR SPECIES 4 = NEQSCH+1
              YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
              IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C             ADJUST THE NEWLY DEFINED VARIABLE AT THIS NODE
              DPENG2(NEQNFL+1,INODE) = RHORPR*YSPEPR(NEQSCH+1)
200        CONTINUE

C          NOW ADJUST THE NUMBER OF EQUATIONS
           YNRTCH = 0.
           NEQNFL = NEQNFL + 1
           NEQSCH = NEQSCH + 1
           NINRCH = NINRCH - 1
        ENDIF

        RETURN
        END
```

HSHEAR

```
        SUBROUTINE HSHEAR
C
        INCLUDE '[.INC] PRECIS.INC/LIST'
        INCLUDE '[.INC] PARMV2.INC/LIST'
        INCLUDE '[.INC] CHCOMN.INC/LIST'
        INCLUDE '[.INC] E2COMN.INC/LIST'
        INCLUDE '[.INC] FLCOMN.INC/LIST'
        INCLUDE '[.INC] G2COMN.INC/LIST'
        INCLUDE '[.INC] IOCOMN.INC/LIST'
        INCLUDE '[.INC] PRCOMN.INC/LIST'
```

```
C
C*********************************************************************

C       THIS SUBROUTINE INITIALIZES THE DEPENDENT VARIABLES FOR SHEAR
C       FLOW.  THE VALUES NEEDED AT THE INTERNAL POINTS ARE THE
C       PROPERTIES OF THIS MIXTURE, I.E., TEMPERATURE, PRESSURE, MACH NO.,
C       EQUIVALENCE RATIO, AND THE ANGLE OF INJECTION.  ALSO NEEDED IS THE
C       TOTAL NUMBER OF INJECTION POINTS AND THE ACTUAL NODE NUMBERS.

C*********************************************************************
C
C       READ THE FOLLOWING FUEL QUANTITIES
C            TEMPEF  : FUEL TEMPERATURE IN DEGREE K
C            PRESSF  : FUEL PRESSURE IN PASCALS
C            AMACHF  : FUEL MACH NUMBER
C            ANGLEF  : ANGLE OF INJECTION IN DEGREES
C            YSPEPR  : MASS FRACTION OF ALL SPECIES
C            NINJEC  : NUMBER OF CELLS WITH FUEL INJECTION (INPUT THIS
C                      AS NEGATIVE IF BOUNDARY NODES ARE NOT TO BE SET)
C            INODE   : CELLS WHERE FUEL IS INJECTED
C
       READ (JREADS,*) TEMPEF
       READ (JREADS,*) PRESSF
       READ (JREADS,*) AMACHF
       READ (JREADS,*) ANGLEF
       DO IQ = 1, NSPECH
          READ (JREADS,*) YSPEPR(IQ)
       ENDDO
       READ (JREADS,*) NINJEC
C
C       COMPUTE THE ANGLE IN RADIANS
       ANGLEF = ANGLEF*3.141592654/180.
C
C       DETERMINE THE MOLECULAR MASS AND OTHER QUANTITIES FOR THIS MIXTURE
C
       DO 5 IS = 1, NSPECH
           SYSBMS = SYSBMS + YSPEPR(IS)*RAMWCH(IS)
           SYSHFE = SYSHFE + YSPEPR(IS)*FMHTCH(IS)
           SYSCPE = SYSCPE + YSPEPR(IS)*SPCPCH(IS)
           BIGAM  = BIGAM  + YSPEPR(IS)*SPBSCH(IS)
5      CONTINUE

       UGASCO = UGASFL*SYSBMS

C      DETERMINE THE DIMENSIONLESS DENSITY OF THE FUEL MIXTURE
       RHOF   = PRESSF/(UGASCO*TEMPEF*RHORFL)

C      DETERMINE THE DIMENSIONLESS PRESSURE OF THE FUEL MIXTURE
       PRESSF = PRESSF/PRESFL
C
C      DETERMINE GAMMA FOR THIS MIXTURE
       BIGAMT = BIGAM*TEMPEF
       SYSCVE = SYSCPE + BIGAMT - UGASFL*SYSBMS
       GAMMAE = (SYSCPE + BIGAMT)/SYSCVE

C      DETERMINE THE OVERALL DIMENSIONLESS VELOCITY OF THE FUEL
       VELOF  = AMACHF*SQRT(GAMMAE*PRESSF/RHOF)
```

```
      UCOMPF = VELOF*COS(ANGLEF)
      VCOMPF = VELOF*SIN(ANGLEF)
      VEL02I = UCOMPF*UCOMPF + VCOMPF*VCOMPF

C     DETERMINE THE ENERGY TERM
      BEE    = SYSHFE + (TEMPEF-TREFCH)*SYSCPE - UGASFL*TEMPEF*SYSBMS
     1                 + 0.5*(TEMPEF*TEMPEF-TREFCH*TREFCH)*BIGAM
      BEE    = BEE/FMREFL + 0.5*VEL02I
C
      TEMPEF = TEMPEF/TREFFL

      DO 30 JCELL = 1, ABS(NINJEC)
C        READ THE NODE NUMBER FOR THIS VALUE
         READ (JREADS,*) INODEF

         if (inodef .eq. 1) then
            write(6,*) ' rho', DPENG2(1,INODEF),RHOF
            write(6,*) ' u  ', DPENG2(2,INODEF),RHOF*ucompf
            write(6,*) ' bee', DPENG2(4,INODEF),RHOF*bee
            write(6,*) ' prs', presG2(INODEF),pressf
            write(6,*) ' tmp', tempG2(INODEF),tempef
         endif

C        SET THE DEPENDENT VARIABLES AT THIS NODE
         DPENG2(1,INODEF) = RHOF
         DPENG2(2,INODEF) = RHOF*UCOMPF
         DPENG2(3,INODEF) = RHOF*VCOMPF
         DPENG2(4,INODEF) = BEE*RHOF

         DO 10 JS = NEQBAS+1, NEQNFL
            IS             = JS - NEQBAS
            DPENG2(JS,INODEF) = RHOF*YSPEPR(IS)
10       CONTINUE

         PRESG2(INODEF) = PRESSF
         TEMPG2(INODEF) = TEMPEF

C        SET THE BOUNDARY CONDITION POINTER

         IF (NINJEC .LT. 0) GOTO 30
         DO 20 JS = 1, NBNDG2
            IF (IBNDG2(1,JS) .EQ. INODEF) THEN
               IBNDG2(5,JS) = 2
               GOTO 30
            ENDIF
20       CONTINUE

         NBNDG2           = NBNDG2 + 1
         IBNDG2(1,NBNDG2) = INODEF
         IBNDG2(2,NBNDG2) = 0
         IBNDG2(3,NBNDG2) = 0
         IBNDG2(4,NBNDG2) = 0
         IBNDG2(5,NBNDG2) = 2
30    CONTINUE
C
C     FOR ROGERS AND CHINITZ MODEL THE NUMBER OF EQUATIONS MUST
C     BE ADJUSTED
```

```
      IF (KROGER .EQ. 1 .AND. NINRCH .GT. 0) THEN
         DO 200 INODE = 1, NNODG2

C           COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
            SUMY   = 0.
            YUPPER = 1. - YNRTCH
            RHORPR = DPENG2(1,INODE)

            DO 190 IS = 1, NEQSCH
               JS         = NEQBAS + IS
               YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
               IF (YSPEPR(IS) .LT. 0.) THEN
                 YSPEPR(IS)       = 0.
                 DPENG2(JS,INODE) = 0.
               ENDIF
               IF (YSPEPR(IS) .GT. 1.) THEN
                 YSPEPR(IS)       = 1.
                 DPENG2(JS,INODE) = YUPPER*RHORPR
               ENDIF
               SUMY       = SUMY + YSPEPR(IS)
190          CONTINUE
C           THE FOLLOWING IS FOR SPECIES 4 = NEQSCH+1
            YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
            IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C           ADJUST THE NEWLY DEFINED VARIABLE AT THIS NODE
            DPENG2(NEQNFL+1,INODE) = RHORPR*YSPEPR(NEQSCH+1)
200      CONTINUE

C        NOW ADJUST THE NUMBER OF EQUATIONS
         YNRTCH = 0.
         NEQNFL = NEQNFL + 1
         NEQSCH = NEQSCH + 1
         NINRCH = NINRCH - 1
      ENDIF

      RETURN
      END
```

LHINI2

```
      SUBROUTINE LHINI2
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
C
C*****************************************************************
C
C     THIS SUBROUTINE INITIALIZES THE CHCOMN COMMON BLOCK FOR A
```

```
C       LIGHT HILL GAS.  IT IS ASSUMED THAT THE FOLLOWING QUANTITIES
C       ARE STORED :
C           - PHI     IN PREFCH(1)
C             ETA     IN EXPFCH(1)
C             THETAD IN ENEFCH(1)
C             RHOD   IN PREBCH(1).
C
C*********************************************************************
C
        IF (KROGER .NE. 2) RETURN

        PHI       = PREFCH(1)
        ETA       = EXPFCH(1)
        THETD     = ENEFCH(1)
        RHOD     = PREBCH(1)
        TOETA     = TREFFL**ETA
        UNITCF    = UREFFL/(TOETA*RHORFL*DISTFL)
        CFBMA     = UNITCF*PHI
        CF        = CFBMA*AMWTCH(1)
        PREFCH(1) = LOG(CFBMA)
        PREBCH(1) = LOG(0.5*CF/RHOD)
        PREECH(1) = LOG(2.*RHOD/AMWTCH(1))
        TREFCH    = 0.
        EXPFCH(1) = ETA
        EXPBCH(1) = ETA
        EXPECH(1) = 0.
        ENEFCH(1) = THETD
        ENEBCH(1) = 0.
        ENEECH(1) = THETD
        RGASA2    = SPCVCH(1)/3.
        SPCVCH(2) = SPCVCH(1)
C       FMHTCH(1) = RGASA2*THETD*AMWTCH(1)
        FMHTCH(1) = RGASA2*THETD
        FMHTCH(2) = 0.
        APASKY(1) = PHI
        APASKY(2) = RHOD
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGFL .NE. 4 .AND. IDBGFL .LT. 1000)  RETURN

        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) PHI, ETA, THETD, RHOD, CF, RGASA2,
     1                     SPCVCH(1), FMHTCH(1), PREFCH(1), PREBCH(1),
     2                     PREECH(1), EXPFCH(1), ENEFCH(1), ENEECH(1)
C
C       ----------------
C       FORMAT STATEMENTS
C       ----------------
C
1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM LHINI2' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT(5X,'PHI     = ', G14.5, 10X, 'ETA    = ', G14.5/
     1         5X,'THETAD = ', G14.5, 10X, 'RHOD  = ', G14.5/
```

```
     2         5X,'CF      = ', G14.5, 10X, 'RGASA2 = ', G14.5/
     3         5X,'CV_A    = ', G14.5, 10X, 'HTFMA  = ', G14.5/
     4       - 5X,'PRE-Af  = ', G14.5, 10X, 'PRE-Ab = ', G14.5/
     5         5X,'PRE-Ae  = ', G14.5, 10X, 'EXP-f  = ', G14.5/
     6         5X,'ENERGYF = ', G14.5, 10X, 'ENERGYE= ', G14.5/)
C
        RETURN
        END
```

M2AREA

```
        SUBROUTINE M2AREA (KONTRL)

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'M2COMN.INC'

C***********************************************************************

C       THIS SUBROUTINE SETS UP THE VOLUME (AREA), PERIMETER, AND THE
C       METRICS FOR THE SOLVER ROUTINE

C***********************************************************************

        IF (KONTRL .GT. 0) GOTO 20
C
C       SETUP THE METRICES ETC FOR EACH CELL IN THE WHOLE SPATIAL DOMAIN
C
        DO 10 ICELL = 1, NCELG2

C          SET UP NODE POINTERS FOR THIS CELL

           KSW = ICELG2( 2,ICELL)
           KSE = ICELG2( 4,ICELL)
           KNE = ICELG2( 6,ICELL)
           KNW = ICELG2( 8,ICELL)

C          --------
C          GEOMETRY
C          --------
C
C          GEOMETRY OF ALL CELL CORNERS

           XSW = GEOMG2(1,KSW)
           YSW = GEOMG2(2,KSW)

           XSE = GEOMG2(1,KSE)
           YSE = GEOMG2(2,KSE)

           XNE = GEOMG2(1,KNE)
           YNE = GEOMG2(2,KNE)

           XNW = GEOMG2(1,KNW)
```

```
            YNW = GEOMG2(2,KNW)

C           COMPUTE THE DISTANCES OF CELL FACES SO THAT ITS PERIMETER
C           CAN BE DETERMINED

            DXS = XSE - XSW
            DYS = YSE - YSW

            DXE = XNE - XSE
            DYE = YNE - YSE

            DXN = XNW - XNE
            DYN = YNW - YNE

            DXW = XSW - XNW
            DYW = YSW - YNW

            DDS = SQRT ( DXS*DXS + DYS*DYS )
            DDE = SQRT ( DXE*DXE + DYE*DYE )
            DDN = SQRT ( DXN*DXN + DYN*DYN )
            DDW = SQRT ( DXW*DXW + DYW*DYW )

            PERIM2(ICELL) = DDS + DDE + DDN + DDW

C           COMPUTE THE PROJECTIONS OF CELL FACES AND THE SIZE OF THE CELL

            DXEWM2(ICELL) = 0.5*( XNE + XSE - XNW - XSW )
            DYEWM2(ICELL) = 0.5*( YNE + YSE - YNW - YSW )
            DXNSM2(ICELL) = 0.5*( XNW + XNE - XSW - XSE )
            DYNSM2(ICELL) = 0.5*( YNW + YNE - YSW - YSE )

C           THE CELL VOLUME

            DVOL = 0.5*( (XSE-XNW)*(YNE-YSW) - (YSE-YNW)*(XNE-XSW) )

C           RECIPROCAL OF THE CELL VOLUME

            RVOLM2(ICELL) = 1./DVOL

10      CONTINUE

        RETURN
C
C       SET UP THE METRICS ETC FOR A SPECIFIED CELL

20      ICELL = KONTRL

C       SET UP NODE POINTERS FOR THIS CELL

        KSW = ICELG2( 2,ICELL)
        KSE = ICELG2( 4,ICELL)
        KNE = ICELG2( 6,ICELL)
        KNW = ICELG2( 8,ICELL)

C       --------
C       GEOMETRY
C       --------
```

```
C
C       GEOMETRY OF ALL CELL CORNERS

        XSW = GEOMG2(1,KSW)
        YSW = GEOMG2(2,KSW)

        XSE = GEOMG2(1,KSE)
        YSE = GEOMG2(2,KSE)

        XNE = GEOMG2(1,KNE)
        YNE = GEOMG2(2,KNE)

        XNW = GEOMG2(1,KNW)
        YNW = GEOMG2(2,KNW)

C       COMPUTE THE DISTANCES OF CELL FACES SO THAT ITS PERIMETER
C       CAN BE DETERMINED

        DXS = XSE - XSW
        DYS = YSE - YSW

        DXE = XNE - XSE
        DYE = YNE - YSE

        DXN = XNW - XNE
        DYN = YNW - YNE

        DXW = XSW - XNW
        DYW = YSW - YNW

        DDS = SQRT ( DXS*DXS + DYS*DYS )
        DDE = SQRT ( DXE*DXE + DYE*DYE )
        DDN = SQRT ( DXN*DXN + DYN*DYN )
        DDW = SQRT ( DXW*DXW + DYW*DYW )

        PERIM2(ICELL) = DDS + DDE + DDN + DDW

C       COMPUTE THE PROJECTIONS OF CELL FACES AND THE SIZE OF THE CELL

        DXEWM2(ICELL) = 0.5*( XNE + XSE - XNW - XSW )
        DYEWM2(ICELL) = 0.5*( YNE + YSE - YNW - YSW )
        DXNSM2(ICELL) = 0.5*( XNW + XNE - XSW - XSE )
        DYNSM2(ICELL) = 0.5*( YNW + YNE - YSW - YSE )

C       THE CELL VOLUME

        DVOL = 0.5*( (XSE-XNW)*(YNE-YSW) - (YSE-YNW)*(XNE-XSW) )

C       RECIPROCAL OF THE CELL VOLUME

        RVOLM2(ICELL) = 1./DVOL

        RETURN
        END
```

NODIT2

```
      SUBROUTINE NODIT2

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'TICOMN.INC'
      LOGICAL ZERNOD

C*********************************************************************

C     THIS SUBROUTINE CORRECTS THE TIME-STEPS AT NODITS IF NECESSARY.
C     NODIT IS THE ACRONYM FOR "NODE OF DIFFERENT TIME-STEPS".
C     A FACTOR DIFFERENCE OF ONLY TWO OR FOUR IS ALLOWED BETWEEN
C     ADJACENT CELLS.

C*********************************************************************

      IF (NMAXTI .LE. 2) RETURN

C     SET THE COUNTER FOR THE NUMBER OF NODITS

      NNODEM = 0

C     SET THE MAXIMUM FACTOR

      LCRAT = 4
C
C     ----------------------
C     INTERIOR INITIAL NODES
C     ----------------------
C
C     STEP THROUGH ALL THE NODES AND FIND CELL TIMESTEPS AND NODITS

      DO 10 INODE = 1, NNODG2

         NBSW   = NEIBG2(1,INODE)
         NBSE   = NEIBG2(2,INODE)
         NBNE   = NEIBG2(3,INODE)
         NBNW   = NEIBG2(4,INODE)

C        MAKE SURE ALL THE FOUR CORNER CELLS EXIST; I.E., MAKE SURE
C        THAT THE NODE UNDER CONSIDERATION IS NOT A BOUNDARY NODE
C        BOUNDARY NODES WILL BE HANDLED SEPERATELY

         ZERNOD = NBSW.EQ.0 .OR. NBSE.EQ.0 .OR.
     1            NBNE.EQ.0 .OR. NBNW.EQ.0

         IF (.NOT. ZERNOD) THEN

            DTSW  = CELLTI(NBSW)
            DTSE  = CELLTI(NBSE)
            DTNE  = CELLTI(NBNE)
```

```
      DTNW  = CELLTI(NBNW)
      DTMAX = MAX (DTSW,DTSE,DTNE,DTNW)
      DTMIN = MIN (DTSW,DTSE,DTNE,DTNW)
      IFACT = NINT (DTMAX/DTMIN)
      IF (IFACT .LE. LCRAT) GO TO 10

C     FIND THE MAXIMUM ALLOWABLE CELL TIME STEP

      DTMAX = LCRAT*DTMIN

C     FIND THE CELL WHICH EXCEEDS THE LIMIT  AND SAVE THE
C     APPROPRAITE NODES (REMAINING THREE)
C     CHECK THE SOUTH-WEST NEIGHBOUR CELL

      IF (DTSW .GT. DTMAX) THEN
        CELLTI(NBSW)     = DTMAX
        MRKCA2(NNODEM+1) = ICELG2(2,NBSW)
        MRKCA2(NNODEM+2) = ICELG2(4,NBSW)
        MRKCA2(NNODEM+3) = ICELG2(8,NBSW)
        NNODEM           = NNODEM + 3
        DTSE             = CELLTI(NBSE)
        DTNW             = CELLTI(NBNW)
      ENDIF
C
C     REASSIGNMENT OF THE CELL TIMESTEPS IN THE ABOVE IF-THEN
C     BLOCK IS NEEDED TO GUARD AGAINST THE CASE WHEN A NODE HAS
C     LESS THAN 4 NEIGHBOURING NODES (E.G., WHEN NBSE = NBSW)
C
C     CHECK THE SOUTH-EAST NEIGHBOUR CELL

      IF (DTSE .GT. DTMAX) THEN
        CELLTI(NBSE)     = DTMAX
        MRKCA2(NNODEM+1) = ICELG2(2,NBSE)
        MRKCA2(NNODEM+2) = ICELG2(4,NBSE)
        MRKCA2(NNODEM+3) = ICELG2(6,NBSE)
        NNODEM           = NNODEM + 3
        DTNE             = CELLTI(NBNE)
      ENDIF

C     CHECK THE NORTH-EAST NEIGHBOUR CELL

      IF (DTNE .GT. DTMAX) THEN
        CELLTI(NBNE)     = DTMAX
        MRKCA2(NNODEM+1) = ICELG2(4,NBNE)
        MRKCA2(NNODEM+2) = ICELG2(6,NBNE)
        MRKCA2(NNODEM+3) = ICELG2(8,NBNE)
        NNODEM           = NNODEM + 3
        DTNW             = CELLTI(NBNW)
      ENDIF

C     CHECK THE NORTH-WEST NEIGHBOUR CELL

      IF (DTNW .GT. DTMAX) THEN
        CELLTI(NBNW)     = DTMAX
        MRKCA2(NNODEM+1) = ICELG2(2,NBNW)
        MRKCA2(NNODEM+2) = ICELG2(6,NBNW)
        MRKCA2(NNODEM+3) = ICELG2(8,NBNW)
```

```
              NNODEM           = NNODEM + 3
            ENDIF

          ENDIF

10      CONTINUE
C
C       ---------------------
C       BOUNDARY INITIAL NODES
C       ---------------------
C
C       NOW CHECK ALL THE BOUNDARY NODES; FIRST SET THE COUNTER FOR
C       THE NUMBER OF BOUNDARY NODES WHICH ARE ALSO NODIT'S

        NBODEM = 0

        DO 20 INBND = 1, NBNDG2

           NBONE  = IBNDG2(2,INBND)
           NBTWO  = IBNDG2(3,INBND)

C          ATLEAST ONE OF THE ABOVE CELLS MUST BE NON-ZERO

           ZERNOD = NBONE.EQ.0 .OR. NBTWO.EQ.0

           IF (.NOT. ZERNOD) THEN

              DTONE = CELLTI(NBONE)
              DTTWO = CELLTI(NBTWO)
              DTMAX = MAX (DTONE,DTTWO)
              DTMIN = MIN (DTONE,DTTWO)
              IFACT = NINT (DTMAX/DTMIN)
              IF (IFACT .LE. LCRAT) GO TO 20
              DTMAX = LCRAT*DTMIN

C             CHECK THE SOUTHERN EDGE

              IF (IBNDG2(4,INBND) .EQ. 3) THEN
                IF (DTONE .GT. DTMAX) THEN
                   CELLTI(NBONE)    = DTMAX
                   MRKCA2(NNODEM+1) = ICELG2(6,NBONE)
                   MRKCA2(NNODEM+2) = ICELG2(8,NBONE)
                   WORKA2(NBODEM+1) = ICELG2(2,NBONE)
                   NNODEM           = NNODEM + 2
                   NBODEM           = NBODEM + 1
                ENDIF
                IF (DTTWO .GT. DTMAX) THEN
                   CELLTI(NBTWO)    = DTMAX
                   MRKCA2(NNODEM+1) = ICELG2(6,NBTWO)
                   MRKCA2(NNODEM+2) = ICELG2(8,NBTWO)
                   WORKA2(NBODEM+1) = ICELG2(4,NBTWO)
                   NNODEM           = NNODEM + 2
                   NBODEM           = NBODEM + 1
                ENDIF
              ENDIF

C             CHECK THE EASTERN EDGE
```

```
      IF (IBNDG2(4,INBND) .EQ. 5) THEN
        IF (DTONE .GT. DTMAX) THEN
           CELLTI(NBONE)    = DTMAX
           MRKCA2(NNODEM+1) = ICELG2(8,NBONE)
           MRKCA2(NNODEM+2) = ICELG2(2,NBONE)
           WORKA2(NBODEM+1) = ICELG2(4,NBONE)
           NNODEM           = NNODEM + 2
           NBODEM           = NBODEM + 1
        ENDIF
        IF (DTTWO .GT. DTMAX) THEN
           CELLTI(NBTWO)    = DTMAX
           MRKCA2(NNODEM+1) = ICELG2(8,NBTWO)
           MRKCA2(NNODEM+2) = ICELG2(2,NBTWO)
           WORKA2(NBODEM+1) = ICELG2(6,NBTWO)
           NNODEM           = NNODEM + 2
           NBODEM           = NBODEM + 1
        ENDIF
      ENDIF

C     CHECK THE NORTHERN EDGE

      IF (IBNDG2(4,INBND) .EQ. 7) THEN
        IF (DTONE .GT. DTMAX) THEN
           CELLTI(NBONE)    = DTMAX
           MRKCA2(NNODEM+1) = ICELG2(2,NBONE)
           MRKCA2(NNODEM+2) = ICELG2(4,NBONE)
           WORKA2(NBODEM+1) = ICELG2(6,NBONE)
           NNODEM           = NNODEM + 2
           NBODEM           = NBODEM + 1
        ENDIF
        IF (DTTWO .GT. DTMAX) THEN
           CELLTI(NBTWO)    = DTMAX
           MRKCA2(NNODEM+1) = ICELG2(2,NBTWO)
           MRKCA2(NNODEM+2) = ICELG2(4,NBTWO)
           WORKA2(NBODEM+1) = ICELG2(8,NBTWO)
           NNODEM           = NNODEM + 2
           NBODEM           = NBODEM + 1
       ENDIF
      ENDIF

C     CHECK THE WESTERN EDGE

      IF (IBNDG2(4,INBND) .EQ. 9) THEN
        IF (DTONE .GT. DTMAX) THEN
           CELLTI(NBONE)    = DTMAX
           MRKCA2(NNODEM+1) = ICELG2(4,NBONE)
           MRKCA2(NNODEM+2) = ICELG2(6,NBONE)
           WORKA2(NBODEM+1) = ICELG2(8,NBONE)
           NNODEM           = NNODEM + 2
           NBODEM           = NBODEM + 1
        ENDIF
        IF (DTTWO .GT. DTMAX) THEN
           CELLTI(NBTWO)    = DTMAX
           MRKCA2(NNODEM+1) = ICELG2(4,NBTWO)
           MRKCA2(NNODEM+2) = ICELG2(6,NBTWO)
           WORKA2(NBODEM+1) = ICELG2(2,NBTWO)
```

```
                NNODEM          = NNODEM + 2
                NBODEM          = NBODEM + 1
             ENDIF
           ENDIF

         ENDIF

20     CONTINUE
C
C      ERROR CONDITION
C
30     IF (NNODEM .GT. MCELG2) THEN
         ZER1 = NNODEM
         ZER2 = MCELG2
         CALL   PSWRTU (JPNTWR)
         CALL ERRORM (44,'NODIT2','NNODEM',ZER1,'MCELG2',ZER2,JPRINT,
     1     'NUMBER OF NODITS EXCEEDS LIMIT; PSWRTU WRITTEN')
       ENDIF
C
C      SEE IF EXIT CONDITION IS MET
C
       IF (NNODEM .EQ. 0 .AND. NBODEM .EQ. 0) RETURN
C
C      ----------------------
C      NEXT SET INTERIOR NODES
C      ----------------------
C
C      PROCESS ALL THE PREVIOUSLY SAVED NODES

       KNODEM = 0

       DO 40 JNODE = 1, NNODEM

          INODE  = MRKCA2(JNODE)
          NBSW   = NEIBG2(1,INODE)
          NBSE   = NEIBG2(2,INODE)
          NBNE   = NEIBG2(3,INODE)
          NBNW   = NEIBG2(4,INODE)
          ZERNOD = NBSW.EQ.0 .OR. NBSE.EQ.0 .OR.
     1             NBNE.EQ.0 .OR. NBNW.EQ.0

          IF (ZERNOD) THEN

             NBODEM = NBODEM + 1
             WORKA2(NBODEM) = INODE

          ELSE

             DTSW  = CELLTI(NBSW)
             DTSE  = CELLTI(NBSE)
             DTNE  = CELLTI(NBNE)
             DTNW  = CELLTI(NBNW)
             DTMAX = MAX (DTSW,DTSE,DTNE,DTNW)
             DTMIN = MIN (DTSW,DTSE,DTNE,DTNW)
             IFACT = NINT (DTMAX/DTMIN)
             IF (IFACT .LE. LCRAT) GO TO 40
             DTMAX = LCRAT*DTMIN
```

```
C            CHECK WHICH CELL AGAIN

             IF (DTSW .GT. DTMAX) THEN
               CELLTI(NBSW)     = DTMAX
               MRKDA2(KNODEM+1) = ICELG2(2,NBSW)
               MRKDA2(KNODEM+2) = ICELG2(4,NBSW)
               MRKDA2(KNODEM+3) = ICELG2(8,NBSW)
               KNODEM           = KNODEM + 3
               DTSE             = CELLTI(NBSE)
               DTNW             = CELLTI(NBNW)
             ENDIF
C
             IF (DTSE .GT. DTMAX) THEN
               CELLTI(NBSE)     = DTMAX
               MRKDA2(KNODEM+1) = ICELG2(2,NBSE)
               MRKDA2(KNODEM+2) = ICELG2(4,NBSE)
               MRKDA2(KNODEM+3) = ICELG2(6,NBSE)
               KNODEM           = KNODEM + 3
               DTNE             = CELLTI(NBNE)
             ENDIF
C
             IF (DTNE .GT. DTMAX) THEN
               CELLTI(NBNE)     = DTMAX
               MRKDA2(KNODEM+1) = ICELG2(4,NBNE)
               MRKDA2(KNODEM+2) = ICELG2(6,NBNE)
               MRKDA2(KNODEM+3) = ICELG2(8,NBNE)
               KNODEM           = KNODEM + 3
               DTNW             = CELLTI(NBNW)
             ENDIF
C
             IF (DTNW .GT. DTMAX) THEN
               CELLTI(NBNW)     = DTMAX
               MRKDA2(KNODEM+1) = ICELG2(2,NBNW)
               MRKDA2(KNODEM+2) = ICELG2(6,NBNW)
               MRKDA2(KNODEM+3) = ICELG2(8,NBNW)
               KNODEM           = KNODEM + 3
             ENDIF

          ENDIF

40       CONTINUE
C
C        RESET THE NEXT SET INTERIOR NODE SET; BECAUSE THE PREVIOUS
C        SET HAS SERVED IT'S PURPOSE
C
         NNODEM = KNODEM

         DO 50 INODE = 1, NNODEM
            MRKCA2(INODE) = MRKDA2(INODE)
50       CONTINUE
C
C        ----------------------
C        NEXT SET BOUNDARY NODES
C        ----------------------
C
C        PROCESS ALL THE PREVIOUSLY SAVED BOUNDARY NODES
```

```
      KBODEM = 0

      DO 80 JNODE = 1, NBODEM

         JNBND  = NINT ( WORKA2(JNODE) )
         DO 60 IBOUND = 1, NBNDG2
           IF (IBNDG2(1,IBOUND) .EQ. JNBND) THEN
              INBND = IBOUND
              GOTO 70
           ENDIF
60       CONTINUE

70       NBONE  = IBNDG2(2,INBND)
         NBTWO  = IBNDG2(3,INBND)
         ZERNOD = NBONE.EQ.0 .OR. NBTWO.EQ.0

         IF (.NOT. ZERNOD) THEN

            DTONE = CELLTI(NBONE)
            DTTWO = CELLTI(NBTWO)
            DTMAX = MAX (DTONE,DTTWO)
            DTMIN = MIN (DTONE,DTTWO)
            IFACT = NINT (DTMAX/DTMIN)
            IF (IFACT .LE. LCRAT) GO TO 80
            DTMAX = LCRAT*DTMIN
C
            IF (IBNDG2(4,INBND) .EQ. 3) THEN
              IF (DTONE .GT. DTMAX) THEN
                 CELLTI(NBONE)    = DTMAX
                 MRKCA2(KNODEM+1) = ICELG2(6,NBONE)
                 MRKCA2(KNODEM+2) = ICELG2(8,NBONE)
                 CHNGA2(KBODEM+1) = ICELG2(2,NBONE)
                 KNODEM           = KNODEM + 2
                 KBODEM           = KBODEM + 1
              ENDIF
              IF (DTTWO .GT. DTMAX) THEN
                 CELLTI(NBTWO)    = DTMAX
                 MRKCA2(KNODEM+1) = ICELG2(6,NBTWO)
                 MRKCA2(KNODEM+2) = ICELG2(8,NBTWO)
                 CHNGA2(KBODEM+1) = ICELG2(4,NBTWO)
                 KNODEM           = KNODEM + 2
                 KBODEM           = KBODEM + 1
              ENDIF
            ENDIF
C
            IF (IBNDG2(4,INBND) .EQ. 5) THEN
              IF (DTONE .GT. DTMAX) THEN
                 CELLTI(NBONE)    = DTMAX
                 MRKCA2(KNODEM+1) = ICELG2(8,NBONE)
                 MRKCA2(KNODEM+2) = ICELG2(2,NBONE)
                 CHNGA2(KBODEM+1) = ICELG2(4,NBONE)
                 KNODEM           = KNODEM + 2
                 KBODEM           = KBODEM + 1
              ENDIF
              IF (DTTWO .GT. DTMAX) THEN
                 CELLTI(NBTWO)    = DTMAX
```

```
                MRKCA2(KNODEM+1) = ICELG2(8,NBTWO)
                MRKCA2(KNODEM+2) = ICELG2(2,NBTWO)
                CHNGA2(KBODEM+1) = ICELG2(6,NBTWO)
                KNODEM           = KNODEM + 2
                KBODEM           = KBODEM + 1
              ENDIF
            ENDIF
C
            IF (IBNDG2(4,INBND) .EQ. 7) THEN
              IF (DTONE .GT. DTMAX) THEN
                CELLTI(NBONE)    = DTMAX
                MRKCA2(KNODEM+1) = ICELG2(2,NBONE)
                MRKCA2(KNODEM+2) = ICELG2(4,NBONE)
                CHNGA2(KBODEM+1) = ICELG2(6,NBONE)
                KNODEM           = KNODEM + 2
                KBODEM           = KBODEM + 1
              ENDIF
              IF (DTTWO .GT. DTMAX) THEN
                CELLTI(NBTWO)    = DTMAX
                MRKCA2(KNODEM+1) = ICELG2(2,NBTWO)
                MRKCA2(KNODEM+2) = ICELG2(4,NBTWO)
                CHNGA2(KBODEM+1) = ICELG2(8,NBTWO)
                KNODEM           = KNODEM + 2
                KBODEM           = KBODEM + 1
              ENDIF
            ENDIF
C
            IF (IBNDG2(4,INBND) .EQ. 9) THEN
              IF (DTONE .GT. DTMAX) THEN
                CELLTI(NBONE)    = DTMAX
                MRKCA2(KNODEM+1) = ICELG2(4,NBONE)
                MRKCA2(KNODEM+2) = ICELG2(6,NBONE)
                CHNGA2(KBODEM+1) = ICELG2(8,NBONE)
                KNODEM           = KNODEM + 2
                KBODEM           = KBODEM + 1
              ENDIF
              IF (DTTWO .GT. DTMAX) THEN
                CELLTI(NBTWO)    = DTMAX
                MRKCA2(KNODEM+1) = ICELG2(4,NBTWO)
                MRKCA2(KNODEM+2) = ICELG2(6,NBTWO)
                CHNGA2(KBODEM+1) = ICELG2(2,NBTWO)
                KNODEM           = KNODEM + 2
                KBODEM           = KBODEM + 1
              ENDIF
            ENDIF

          ENDIF

80      CONTINUE
C
C       RESET THE NEXT SET BOUNDARY NODE SET; BECAUSE THE PREVIOUS
C       SET HAS SERVED IT'S PURPOSE
C
        NBODEM = KBODEM

        DO 90 INBND = 1, NBODEM
          WORKA2(INBND) = CHNGA2(INBND)
```

```
90      CONTINUE

        GO TO 30

        END
```

PSRED2

```
        SUBROUTINE PSRED2

        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'CHCOMN.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'FLCOMN.INC'
        INCLUDE 'FRCOMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'H2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'PRCOMN.INC'
        INCLUDE 'TICOMN.INC'
        INCLUDE 'TVCOMN.INC'
        character*80 mtitle2

C***********************************************************************

C       THIS SUBROUTINE READS ALL THE INFORMATION ABOUT THE POINTER
C       SYSTEM AND ALL THE OTHER ARRAYS FROM UNIT 'JPNTRE'

C***********************************************************************

C       -------------
C       INITIALIZATION
C       -------------

        MCELLP = 10
        MGEOMP = 2
        MBONDP = 5
        MNEIBP = 4

C       INITIALIZE ALL THE INTEGER AND REAL ARRAYS

        DO 10 LC = 1, MCELG2
          KAUXG2(LC) = 0
          ICELA2(LC) = 0
          CHNGA2(LC) = 0.
          MRKCA2(LC) = 0
          MRKDA2(LC) = 0
10      CONTINUE

        DO 20 IN = 1, MNODG2
          WORKA2(IN) = 0.
```

```
          PRESG2(IN) = 0.
          TEMPG2(IN) = 0.
          SIGGE2(IN) = 0.
20      CONTINUE

        DO 30 IR = 1, MREACH
          PREFCH(IR) = 0.
          PREBCH(IR) = 0.
          PREECH(IR) = 0.
          EXPFCH(IR) = 0.
          EXPBCH(IR) = 0.
          EXPECH(IR) = 0.
          ENEFCH(IR) = 0.
          ENEBCH(IR) = 0.
          ENEECH(IR) = 0.
30      CONTINUE

        DO 40 IS = 1, MSPECH
          SPCPCH(IR) = 0.
          SPCVCH(IR) = 0.
          SPBSCH(IR) = 0.
          FMHTCH(IR) = 0.
          YSPECH(IR) = 0.
          AMWTCH(IR) = 0.
          ENTRCH(IR) = 0.
40      CONTINUE

        DO 60 IR = 1, MREACH
           NSRKCH(IR) = 0
           DO 50 IS = 1, MSPECH
              BMIACH(IS,IR) = 0.
              IALPCH(IS,IR) = 0
              IBETCH(IS,IR) = 0
              IALOCH(IS,IR) = 0
              IBTOCH(IS,IR) = 0
              ITABCH(IS,IR) = 0
50         CONTINUE
60      CONTINUE

        DO 80 IQ = 1, MEQNFL
           DO 70 IN = 1, MNODG2
              CHNGE2(IQ,IN) = 0.
              DPENG2(IQ,IN) = 0.
70         CONTINUE
80      CONTINUE

        DO 100 IP = 1, MCELLP
           DO 90 LC = 1, MCELG2
              ICELG2(IP,LC) = 0
90         CONTINUE
100     CONTINUE

        DO 120 IP = 1, MBONDP
           DO 110 IB = 1, MBNDG2
              IBNDG2(IP,IB) = 0
110        CONTINUE
120     CONTINUE
```

```
      DO 140 IP = 1, MGEOMP
         DO 130 IN = 1, MNODG2
            GEOMG2(IP,IN) = 0.
130      CONTINUE
140   CONTINUE

      DO 160 IP = 1, MNEIBP
         DO 150 IN = 1, MNODG2
            NEIBG2(IP,IN) = 0
150      CONTINUE
160   CONTINUE

      DO 180 LV = -MLVLG2, MLVLG2
         DO 170 IP = 1, 3
            ILVLG2(IP,LV) = 0
170      CONTINUE
180   CONTINUE

      DO 190 LV = 1, MUMDH2
         NODEH2(LV) = 0
190   CONTINUE
C
C     ------------------
C     NON-ARRAY INTEGERS
C     ------------------

C     READ ALL THE NON-ARRAY INTEGERS FIRST
C     INTEGERS FORM PARMV2

      READ (JPNTRE,1) NEQNFL, NREACH, NSPECH, NNODG2, NCELG2, NBNDG2,
     1                NLVLG2, NEQBAS, KROGER
1     FORMAT(8I10)

C     INTEGERS FROM A2COMN

      READ (JPNTRE,1) NXTDA2, METHA2, NCELA2, K1ADA2, K2ADA2, MTYPA2,
     1                NPLCA2, IDBGA2, MITRA2, KCHKA2, MTHRA2, KPLTA2,
     2                KMERA2

C     INTEGERS FROM CHCOMN

      READ (JPNTRE,1) IDBGCH, NINRCH, NEQSCH

C     INTEGERS FROM E2COMN

      READ (JPNTRE,1) IDBGE2, MITRE2, KSRTE2, KONVE2, KEQNE2
      NITRE2 = 1

C     INTEGERS FROM FLCOMN

      READ (JPNTRE,1) IDBGFL

C     INTEGERS FROM FRCOMN

      READ (JPNTRE,1) IDBGFR, KPERFR, MCYCFR, NCYCFR
```

```
C       INTEGERS FROM G2COMN

        READ-(JPNTRE,1) IDBGG2, MALVG2, NCRSG2

C       INTEGERS FROM IOCOMN

        READ (JPNTRE,1) JTERMI, JTERMO, JPRINT, JCARDS, JREADI,
     1                  JREADG, JREADC, JREADD, JREADF, JOUTAL,
     2                  JHISTO, JGIVEN, JPNTWR, JDUMY1, JDUMY2,
     3                  JDUMY3, JDUMY4, JDEBUG, JREADS

C       INTEGERS FROM TICOMN

        READ (JPNTRE,1) KTIMTI, NGIVTI, KADPTI, NMAXTI, IMPLTI,
     1                  KFACTI

C       --------------
C       ARRAY INTEGERS
C       --------------

C       INTEGERS FROM A2COMN

        READ (JPNTRE,1) (ICELA2(LC), LC = 1, NCELA2)

C       INTEGERS FROM CHCOMN

        DO 300 IR = 1, NREACH
          READ (JPNTRE,1)  NSRKCH(IR)
          READ (JPNTRE,1) (IALPCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE,1) (IBETCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE,1) (IALOCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE,1) (IBTOCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE,1) (ITABCH(IS,IR), IS = 1, NSPECH)
300     CONTINUE

C       INTEGERS FROM G2COMN

        DO 310 LC = 1, NCELG2
          READ (JPNTRE,1) (ICELG2(IP,LC), IP = 1, MCELLP), KAUXG2(LC)
310     CONTINUE

        DO 320 IB = 1, NBNDG2
          READ (JPNTRE,1) (IBNDG2(IP,IB), IP = 1, MBONDP)
320     CONTINUE

        DO 330 IN = 1, NNODG2
          READ (JPNTRE,1) (NEIBG2(IP,IN), IP = 1, MNEIBP)
330     CONTINUE

        DO 340 LV = -MLVLG2, MLVLG2
          READ (JPNTRE,1) (ILVLG2(IP,LV), IP = 1, 3)
340     CONTINUE

        READ (JPNTRE,1) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

C       ----------------------
C       NON-ARRAY REAL NUMBERS
```

```
C       ---------------------

2       FORMAT(8E15.8)

C       REAL NUMBERS FROM A2COMN

        READ (JPNTRE,2) ALPHA2, BETAA2, GAMMA2, DELTA2, THRDA2, THRCA2

C       REAL NUMBERS FROM CHCOMN

        READ (JPNTRE,2) TREFCH, PRESCH, YNRTCH, TRIGCH

C       REAL NUMBERS FROM E2COMN

        READ (JPNTRE,2) SDELE2, SMAXE2, SMINE2, EPSLE2

C       REAL NUMBERS FROM FLCOMN

        READ (JPNTRE,2) TREFFL, PRESFL, UGASFL, AMCHFL, DISTFL,
     1                  RHORFL, UREFFL, FMREFL, WDREFL, AMWTFL,
     2                  GAMAFL

C       REAL NUMBERS FROM FRCOMN

        READ (JPNTRE,2) RHORFR, UCOMFR, VCOMFR, PRESFR, PBPIFR

C       REAL NUMBERS FROM TICOMN

        READ (JPNTRE,2) CFLNTI, TIMXTI, TIMNTI, EPS1TI, EPSOTI,
     1                  DTCNTI, FCTRTI, ERRMTI

C       READ THE CPU TIME HERE AND SAVE IT

        READ (JPNTRE,2) ZCUM
        WORKA2(3) = ZCUM
        CALL TIMERR (JOUTAL, ZCUM, 'RESTART')

C       ------------------
C       ARRAY REAL NUMBERS
C       ------------------

C       REAL NUMBERS FROM CHCOMN

        READ (JPNTRE,2) (PREFCH(IR), IR = 1, NREACH)
        READ (JPNTRE,2) (PREBCH(IR), IR = 1, NREACH)
        READ (JPNTRE,2) (PREECH(IR), IR = 1, NREACH)

        READ (JPNTRE,2) (EXPFCH(IR), IR = 1, NREACH)
        READ (JPNTRE,2) (EXPBCH(IR), IR = 1, NREACH)
        READ (JPNTRE,2) (EXPECH(IR), IR = 1, NREACH)

        READ (JPNTRE,2) (ENEFCH(IR), IR = 1, NREACH)
        READ (JPNTRE,2) (ENEBCH(IR), IR = 1, NREACH)
        READ (JPNTRE,2) (ENEECH(IR), IR = 1, NREACH)

        READ (JPNTRE,2) (SPCPCH(IS), IS = 1, NSPECH)
        READ (JPNTRE,2) (SPCVCH(IS), IS = 1, NSPECH)
```

```
      READ (JPNTRE,2) (SPBSCH(IS), IS = 1, NSPECH)
      READ (JPNTRE,2) (FMHTCH(IS), IS = 1, NSPECH)
      READ-(JPNTRE,2) (YSPECH(IS), IS = 1, NSPECH)
      READ (JPNTRE,2) (AMWTCH(IS), IS = 1, NSPECH)
      READ (JPNTRE,2) (ENTRCH(IS), IS = 1, NSPECH)
      DO 400 IR = 1, NREACH
        READ (JPNTRE,2) (BMIACH(IS,IR), IS = 1, NSPECH)
400   CONTINUE

C     REAL NUMBERS FROM E2COMN
C     READ (JPNTRE,2) (SIGGE2(IN), IN = 1, NNODG2)

C     REAL NUMBERS FROM FRCOMN

      READ (JPNTRE,2) (DPENFR(IN), IN = 1, MEQNFL)

C     REAL NUMBERS FROM G2COMN

      DO 410 IN = 1, NNODG2
         READ (JPNTRE,2) (DPENG2(IQ,IN), IQ = 1, NEQNFL)
410   CONTINUE

      DO 420 IN = 1, NNODG2
         READ (JPNTRE,2) (GEOMG2(IP,IN), IP = 1, MGEOMP)
420   CONTINUE

      READ (JPNTRE,2) (PRESG2(IN), IN = 1, NNODG2)
      READ (JPNTRE,2) (TEMPG2(IN), IN = 1, NNODG2)

C     REAL NUMBERS FROM PRCOMN

      DO 430 IS = 1, NSPECH
        YSPEPR(IS) = YSPECH(IS)
430   CONTINUE

C     ---------------
C     OTHER VARIABLES
C     ---------------

3     FORMAT(A80)
      READ (JPNTRE,3) MTITLE2
      READ (JPNTRE,2) PHI, RHOD

C     SAVE VALUES SO THAT THEY CAN BE TRANSPORTED TO E2INIO AND WRINI2

      CHNGE2(1,1) = PHI
      CHNGE2(1,2) = RHOD

      CLOSE (JPNTRE)

C     SEE IF TEMPORALLY VARYING CONDITIONS WERE USED
      IF (KPERFR .EQ. 1) THEN
          FLOWTV = PBPIFR
          FREQTV = EPSLE2
          AMPLTV = FLOAT(IDBGFR)/100.
      ENDIF
```

```
      RETURN
      END
```

PSREDU

```
      SUBROUTINE PSREDU

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'H2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'PRCOMN.INC'
      INCLUDE 'TICOMN.INC'
      INCLUDE 'TVCOMN.INC'
      character*80 mtitle2

C*********************************************************************

C     THIS SUBROUTINE READS ALL THE INFORMATION ABOUT THE POINTER
C     SYSTEM AND ALL THE OTHER ARRAYS FROM UNIT 'JPNTRE'

C*********************************************************************

C     --------------
C     INITIALIZATION
C     --------------

      MCELLP = 10
      MGEOMP = 2
      MBONDP = 5
      MNEIBP = 4

C     INITIALIZE ALL THE INTEGER AND REAL ARRAYS

      DO 10 LC = 1, MCELG2
        KAUXG2(LC) = 0
        ICELA2(LC) = 0
        CHNGA2(LC) = 0.
        MRKCA2(LC) = 0
        MRKDA2(LC) = 0
10    CONTINUE

      DO 20 IN = 1, MNODG2
        WORKA2(IN) = 0.
        PRESG2(IN) = 0.
        TEMPG2(IN) = 0.
        SIGGE2(IN) = 0.
```

```
20      CONTINUE

        DO 30 IR = 1, MREACH
          PREFCH(IR) = 0.
          PREBCH(IR) = 0.
          PREECH(IR) = 0.
          EXPFCH(IR) = 0.
          EXPBCH(IR) = 0.
          EXPECH(IR) = 0.
          ENEFCH(IR) = 0.
          ENEBCH(IR) = 0.
          ENEECH(IR) = 0.
30      CONTINUE

        DO 40 IS = 1, MSPECH
          SPCPCH(IR) = 0.
          SPCVCH(IR) = 0.
          SPBSCH(IR) = 0.
          FMHTCH(IR) = 0.
          YSPECH(IR) = 0.
          AMWTCH(IR) = 0.
          ENTRCH(IR) = 0.
40      CONTINUE

        DO 60 IR = 1, MREACH
           NSRKCH(IR) = 0
           DO 50 IS = 1, MSPECH
              BMIACH(IS,IR) = 0.
              IALPCH(IS,IR) = 0
              IBETCH(IS,IR) = 0
              IALOCH(IS,IR) = 0
              IBTOCH(IS,IR) = 0
              ITABCH(IS,IR) = 0
50         CONTINUE
60      CONTINUE

        DO 80 IQ = 1, MEQNFL
           DO 70 IN = 1, MNODG2
              CHNGE2(IQ,IN) = 0.
              DPENG2(IQ,IN) = 0.
70         CONTINUE
80      CONTINUE

        DO 100 IP = 1, MCELLP
           DO 90 LC = 1, MCELG2
              ICELG2(IP,LC) = 0
90         CONTINUE
100     CONTINUE

        DO 120 IP = 1, MBONDP
           DO 110 IB = 1, MBNDG2
              IBNDG2(IP,IB) = 0
110        CONTINUE
120     CONTINUE

        DO 140 IP = 1, MGEOMP
           DO 130 IN = 1, MNODG2
```

```
            GEOMG2(IP,IN) = 0.
130         CONTINUE
140      CONTINUE

         DO 160 IP = 1, MNEIBP
            DO 150 IN = 1, MNODG2
               NEIBG2(IP,IN) = 0
150         CONTINUE
160      CONTINUE

         DO 180 LV = -MLVLG2, MLVLG2
            DO 170 IP = 1, 3
               ILVLG2(IP,LV) = 0
170         CONTINUE
180      CONTINUE

         DO 190 LV = 1, MUMDH2
            NODEH2(LV) = 0
190      CONTINUE
C
C        ------------------
C        NON-ARRAY INTEGERS
C        ------------------

C        READ ALL THE NON-ARRAY INTEGERS FIRST
C        INTEGERS FORM PARMV2

         READ (JPNTRE) NEQNFL, NREACH, NSPECH, NNODG2, NCELG2, NBNDG2,
     1                 NLVLG2, NEQBAS, KROGER

C        INTEGERS FROM A2COMN

         READ (JPNTRE) NXTDA2, METHA2, NCELA2, K1ADA2, K2ADA2, MTYPA2,
     1                 NPLCA2, IDBGA2, MITRA2, KCHKA2, MTHRA2, KPLTA2,
     2                 KMERA2

C        INTEGERS FROM CHCOMN

         READ (JPNTRE) IDBGCH, NINRCH, NEQSCH

C        INTEGERS FROM E2COMN

         READ (JPNTRE) IDBGE2, MITRE2, KSRTE2, KONVE2, KEQNE2
         NITRE2 = 1

C        INTEGERS FROM FLCOMN

         READ (JPNTRE) IDBGFL

C        INTEGERS FROM FRCOMN

         READ (JPNTRE) IDBGFR, KPERFR, MCYCFR, NCYCFR

C        INTEGERS FROM G2COMN

         READ (JPNTRE) IDBGG2, MALVG2, NCRSG2
```

```
C       INTEGERS FROM IOCOMN

        READ -(JPNTRE) JTERMI, JTERMO, JPRINT, JCARDS, JREADI,
     1                 JREADG, JREADC, JREADD, JREADF, JOUTAL,
     2                 JHISTO, JGIVEN, JPNTWR, JDUMY1, JDUMY2,
     3                 JDUMY3, JDUMY4, JDEBUG, JREADS

C       INTEGERS FROM TICOMN

        READ (JPNTRE) KTIMTI, NGIVTI, KADPTI, NMAXTI, IMPLTI,
     1                 KFACTI

C       --------------
C       ARRAY INTEGERS
C       --------------

C       INTEGERS FROM A2COMN

        READ (JPNTRE) (ICELA2(LC), LC = 1, NCELA2)

C       INTEGERS FROM CHCOMN

        DO 300 IR = 1, NREACH
          READ (JPNTRE)  NSRKCH(IR)
          READ (JPNTRE) (IALPCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE) (IBETCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE) (IALOCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE) (IBTOCH(IS,IR), IS = 1, NSPECH)
          READ (JPNTRE) (ITABCH(IS,IR), IS = 1, NSPECH)
300     CONTINUE

C       INTEGERS FROM G2COMN

        DO 310 LC = 1, NCELG2
          READ (JPNTRE) (ICELG2(IP,LC), IP = 1, MCELLP), KAUXG2(LC)
310     CONTINUE

        DO 320 IB = 1, NBNDG2
          READ (JPNTRE) (IBNDG2(IP,IB), IP = 1, MBONDP)
320     CONTINUE

        DO 330 IN = 1, NNODG2
          READ (JPNTRE) (NEIBG2(IP,IN), IP = 1, MNEIBP)
330     CONTINUE

        DO 340 LV = -MLVLG2, MLVLG2
          READ (JPNTRE) (ILVLG2(IP,LV), IP = 1, 3)
340     CONTINUE

        READ (JPNTRE) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

C       ----------------------
C       NON-ARRAY REAL NUMBERS
C       ----------------------

C       REAL NUMBERS FROM A2COMN
```

```
      READ (JPNTRE) ALPHA2, BETAA2, GAMMA2, DELTA2, THRDA2, THRCA2

C     REAL NUMBERS FROM CHCOMN

      READ (JPNTRE) TREFCH, PRESCH, YNRTCH, TRIGCH

C     REAL NUMBERS FROM E2COMN

      READ (JPNTRE) SDELE2, SMAXE2, SMINE2, EPSLE2

C     REAL NUMBERS FROM FLCOMN

      READ (JPNTRE) TREFFL, PRESFL, UGASFL, AMCHFL, DISTFL,
     1              RHORFL, UREFFL, FMREFL, WDREFL, AMWTFL,
     2              GAMAFL

C     REAL NUMBERS FROM FRCOMN

      READ (JPNTRE) RHORFR, UCOMFR, VCOMFR, PRESFR, PBPIFR

C     REAL NUMBERS FROM TICOMN

      READ (JPNTRE) CFLNTI, TIMXTI, TIMNTI, EPS1TI, EPSOTI,
     1              DTCNTI, FCTRTI, ERRMTI

C     READ THE CPU TIME HERE AND SAVE IT

      READ (JPNTRE) ZCUM
      WORKA2(3) = ZCUM
      CALL TIMERR (JOUTAL, ZCUM, 'RESTART')

C     ------------------
C     ARRAY REAL NUMBERS
C     ------------------

C     REAL NUMBERS FROM CHCOMN

      READ (JPNTRE) (PREFCH(IR), IR = 1, NREACH)
      READ (JPNTRE) (PREBCH(IR), IR = 1, NREACH)
      READ (JPNTRE) (PREECH(IR), IR = 1, NREACH)

      READ (JPNTRE) (EXPFCH(IR), IR = 1, NREACH)
      READ (JPNTRE) (EXPBCH(IR), IR = 1, NREACH)
      READ (JPNTRE) (EXPECH(IR), IR = 1, NREACH)

      READ (JPNTRE) (ENEFCH(IR), IR = 1, NREACH)
      READ (JPNTRE) (ENEBCH(IR), IR = 1, NREACH)
      READ (JPNTRE) (ENEECH(IR), IR = 1, NREACH)

      READ (JPNTRE) (SPCPCH(IS), IS = 1, NSPECH)
      READ (JPNTRE) (SPCVCH(IS), IS = 1, NSPECH)
      READ (JPNTRE) (SPBSCH(IS), IS = 1, NSPECH)
      READ (JPNTRE) (FMHTCH(IS), IS = 1, NSPECH)
      READ (JPNTRE) (YSPECH(IS), IS = 1, NSPECH)
      READ (JPNTRE) (AMWTCH(IS), IS = 1, NSPECH)
      READ (JPNTRE) (ENTRCH(IS), IS = 1, NSPECH)
      DO 400 IR = 1, NREACH
```

```
          READ (JPNTRE) (BMIACH(IS,IR), IS = 1, NSPECH)
400     CONTINUE

C       REAL NUMBERS FROM E2COMN
C       READ (JPNTRE) (SIGGE2(IN), IN = 1, NNODG2)

C       REAL NUMBERS FROM FRCOMN

        READ (JPNTRE) (DPENFR(IN), IN = 1, MEQNFL)

C       REAL NUMBERS FROM G2COMN

        DO 410 IN = 1, NNODG2
           READ (JPNTRE) (DPENG2(IQ,IN), IQ = 1, NEQNFL)
410     CONTINUE

        DO 420 IN = 1, NNODG2
           READ (JPNTRE) (GEOMG2(IP,IN), IP = 1, MGEOMP)
420     CONTINUE

        READ (JPNTRE) (PRESG2(IN), IN = 1, NNODG2)
        READ (JPNTRE) (TEMPG2(IN), IN = 1, NNODG2)

C       REAL NUMBERS FROM PRCOMN

        DO 430 IS = 1, NSPECH
          YSPEPR(IS) = YSPECH(IS)
430     CONTINUE

C       ---------------
C       OTHER VARIABLES
C       ---------------

        READ (JPNTRE) MTITLE2
        READ (JPNTRE) PHI, RHOD

C       SAVE VALUES SO THAT THEY CAN BE TRANSPORTED TO E2INIO AND WRINI2

        CHNGE2(1,1) = PHI
        CHNGE2(1,2) = RHOD

        CLOSE (JPNTRE)

C       SEE IF TEMPORALLY VARYING CONDITIONS WERE USED
        IF (KPERFR .EQ. 1) THEN
            FLOWTV = PBPIFR
            FREQTV = EPSLE2
            AMPLTV = FLOAT(IDBGFR)/100.
        ENDIF

        RETURN
        END
```

PSWCOR

```
      SUBROUTINE PSWCOR (JGIVEN)

      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] A2COMN.INC/LIST'
      INCLUDE '[.INC] CHCOMN.INC/LIST'
      INCLUDE '[.INC] E2COMN.INC/LIST'
      INCLUDE '[.INC] FLCOMN.INC/LIST'
      INCLUDE '[.INC] FRCOMN.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] H2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] KYCOMN.INC/LIST'
      INCLUDE '[.INC] PRCOMN.INC/LIST'
      INCLUDE '[.INC] TICOMN.INC/LIST'

C*********************************************************************

C     THIS SUBROUTINE WRITES ALL INFORMATION ABOUT THE COARSE POINTER
C     SYSTEM AND ALL THE OTHER ARRAYS ON UNIT JGIVEN

C*********************************************************************

C     -------------
C     INITIALIZATION
C     -------------

      MCELLP = 10
      MGEOMP = 2
      MBONDP = 5
      MNEIBP = 4
C
C     CORRECT THE TOTAL NUMBER OF CELLS
C
      NCELA2 = ILVLG2(2,0)
      NCELG2 = ILVLG2(2,0)
      NLVLG2 = 0
      MALVG2 = 0
      NCRSG2 = 0
      NPLCA2 = NCELG2
C
C     CORRECT THE CEWIC CELL ARRAY AND THE POINTERS TO EDGE NODES,
C     ALSO DETERMINE THE MAXIMUM NODE
C
      MAXNOD = 0
      DO 100 ICELL = 1, NCELA2
         ICELA2(ICELL) = ICELL
         KSW           = ICELG2(2,ICELL)
         KSE           = ICELG2(4,ICELL)
         KNE           = ICELG2(6,ICELL)
         KNW           = ICELG2(8,ICELL)
         MAXNOD        = MAX (MAXNOD, KSW, KSE, KNE, KNW)
         IF (ICELG2(1,ICELL) .NE. 0) ICELG2(1,ICELL) = 0
         IF (ICELG2(3,ICELL) .NE. 0) ICELG2(3,ICELL) = 0
```

```
          IF (ICELG2(5,ICELL) .NE. 0) ICELG2(5,ICELL) = 0
          IF (ICELG2(7,ICELL) .NE. 0) ICELG2(7,ICELL) = 0
          IF (ICELG2(9,ICELL) .NE. 0) ICELG2(9,ICELL) = 0
100     CONTINUE
C
C       CORRECT THE TOTAL NUMBER OF NODES
C
C       NNODOL = NNODG2
        NNODG2 = MAXNOD
C
C       CORRECT THE TOTAL NUMBER OF BOUNDARY NODES AND THEIR CELL POINTERS
C
        MAXNOD = 0
        DO 130 IBND = 1, NBNDG2
          IF (IBNDG2(1,IBND) .GT. NNODG2) GOTO 130
          MAXNOD = MAXNOD + 1
          IONE = IBNDG2(2,IBND)
          ITWO = IBNDG2(3,IBND)
110       CONTINUE
          IF (IONE .GT. 0) THEN
             IF (ICELG2(10,IONE) .GT. 0) THEN
                IONE = ICELG2(10,IONE)
                GOTO 110
             ENDIF
          ENDIF
120       CONTINUE
          IF (ITWO .GT. 0) THEN
             IF (ICELG2(10,ITWO) .GT. 0) THEN
                ITWO = ICELG2(10,ITWO)
                GOTO 120
             ENDIF
          ENDIF
          IBNDG2(2,IBND) = IONE
          IBNDG2(3,IBND) = ITWO
130     CONTINUE

        NBNDG2 = MAXNOD
C
C       CORRECT THE NEIGHBOUR NODE ARRAY POINTERS
C
        DO 180 IN = 1, NNODG2
          NB1 = NEIBG2(1,IN)
          NB2 = NEIBG2(2,IN)
          NB3 = NEIBG2(3,IN)
          NB4 = NEIBG2(4,IN)
140       CONTINUE
          IF (NB1 .GT. 0) THEN
             IF (ICELG2(10,NB1) .GT. 0) THEN
C               write(6,*) ' nb check',in,nb1,ICELG2(10,NB1)
                NB1 = ICELG2(10,NB1)
                GOTO 140
             ENDIF
          ENDIF
150       CONTINUE
          IF (NB2 .GT. 0) THEN
             IF (ICELG2(10,NB2) .GT. 0) THEN
                NB2 = ICELG2(10,NB2)
```

```
                GOTO 150
              ENDIF
            ENDIF
160         CONTINUE
            IF (NB3 .GT. 0) THEN
               IF (ICELG2(10,NB3) .GT. 0) THEN
                  NB3 = ICELG2(10,NB3)
                  GOTO 160
               ENDIF
            ENDIF
170         CONTINUE
            IF (NB4 .GT. 0) THEN
               IF (ICELG2(10,NB4) .GT. 0) THEN
                  NB4 = ICELG2(10,NB4)
                  GOTO 170
               ENDIF
            ENDIF
            NEIBG2(1,IN) = NB1
            NEIBG2(2,IN) = NB2
            NEIBG2(3,IN) = NB3
            NEIBG2(4,IN) = NB4
180      CONTINUE
C
C        CORRECT THE LEVEL POINTERS
C
         DO 190 LV = 1, MLVLG2
           ILVLG2(1,LV) = NCELG2+1
           ILVLG2(2,LV) = NCELG2
           ILVLG2(3,LV) = 0
190      CONTINUE
C
C        CORRECT THE NBCPG2 ARRAY
C
         IBNSW = 0
         IBNSE = 0
         IBNNE = 0
         IBNNW = 0
         DO 200 IBND = 1, NBNDG2
            IF (IBNDG2(4,IBND) .EQ. 2) THEN
              IF (IBNSW .NE. 0) WRITE(6,210) IBNSW, IBND
              IBNSW = IBND
            ENDIF
            IF (IBNDG2(4,IBND) .EQ. 4) THEN
              IF (IBNSE .NE. 0) WRITE(6,220) IBNSE, IBND
              IBNSE = IBND
            ENDIF
            IF (IBNDG2(4,IBND) .EQ. 6) THEN
              IF (IBNNE .NE. 0) WRITE(6,230) IBNNE, IBND
              IBNNE = IBND
            ENDIF
            IF (IBNDG2(4,IBND) .EQ. 8) THEN
              IF (IBNNW .NE. 0) WRITE(6,240) IBNNW, IBND
              IBNNW = IBND
            ENDIF
200      CONTINUE
210      FORMAT(' MORE THAN ONE SW CORNER',2I5)
220      FORMAT(' MORE THAN ONE SW CORNER',2I5)
```

```
230     FORMAT(' MORE THAN ONE SW CORNER',2I5)
240     FORMAT(' MORE THAN ONE SW CORNER',2I5)
        -
        IF (IBNSW .EQ. 0) WRITE(6,*) ' NO SW CORNER'
        IF (IBNSE .EQ. 0) WRITE(6,*) ' NO SE CORNER'
        IF (IBNNE .EQ. 0) WRITE(6,*) ' NO NE CORNER'
        IF (IBNNW .EQ. 0) WRITE(6,*) ' NO NW CORNER'

        IONE = IBNDG2(2,IBNSW)
        DO 250 IBND = 1, NBNDG2
           IF (IBNDG2(1,IBND) .EQ. ICELG2(8,IONE)) NBCPG2(1,1) = IBND
           IF (IBNDG2(1,IBND) .EQ. ICELG2(4,IONE)) NBCPG2(1,2) = IBND
250     CONTINUE

        IONE = IBNDG2(2,IBNSE)
        DO 260 IBND = 1, NBNDG2
           IF (IBNDG2(1,IBND) .EQ. ICELG2(2,IONE)) NBCPG2(2,1) = IBND
           IF (IBNDG2(1,IBND) .EQ. ICELG2(6,IONE)) NBCPG2(2,2) = IBND
260     CONTINUE

        IONE = IBNDG2(2,IBNNE)
        DO 270 IBND = 1, NBNDG2
           IF (IBNDG2(1,IBND) .EQ. ICELG2(4,IONE)) NBCPG2(3,1) = IBND
           IF (IBNDG2(1,IBND) .EQ. ICELG2(8,IONE)) NBCPG2(3,2) = IBND
270     CONTINUE

        IONE = IBNDG2(2,IBNNW)
        DO 280 IBND = 1, NBNDG2
           IF (IBNDG2(1,IBND) .EQ. ICELG2(6,IONE)) NBCPG2(4,1) = IBND
           IF (IBNDG2(1,IBND) .EQ. ICELG2(2,IONE)) NBCPG2(4,2) = IBND
280     CONTINUE

C       ------------------
C       NON-ARRAY INTEGERS
C       ------------------

C       WRITE ALL THE NON-ARRAY INTEGERS FIRST
C       INTEGERS FORM PARMV2

        WRITE (JGIVEN) NEQNFL, NREACH, NSPECH, NNODG2, NCELG2,
     1                 NBNDG2, NLVLG2, NEQBAS, KROGER

C       INTEGERS FROM A2COMN

        WRITE (JGIVEN) NXTDA2, METHA2, NCELA2, K1ADA2, K2ADA2,
     1                 MTYPA2, NPLCA2, IDBGA2, MITRA2, KCHKA2,
     2                 MTHRA2, KPLTA2, KMERA2

C       INTEGERS FROM CHCOMN

        WRITE (JGIVEN) IDBGCH, NINRCH, NEQSCH

C       INTEGERS FROM E2COMN

        KSRTE2 = 1001
        WRITE (JGIVEN) IDBGE2, MITRE2, KSRTE2, KONVE2, KEQNE2
```

```
C       INTEGERS FROM FLCOMN

        WRITE (JGIVEN) IDBGFL

C       INTEGERS FROM FRCOMN

        WRITE (JGIVEN) IDBGFR, KPERFR, MCYCFR, NCYCFR

C       INTEGERS FROM G2COMN

        WRITE (JGIVEN) IDBGG2, MALVG2, NCRSG2

C       INTEGERS FROM IOCOMN

        WRITE (JGIVEN) JTERMI, JTERMO, JPRINT, JCARDS, JREADI,
     1                 JREADG, JREADC, JREADD, JREADF, JOUTAL,
     2                 JHISTO, JGIVEN, JPNTWR, JDUMY1, JDUMY2,
     3                 JDUMY3, JDUMY4, JDEBUG, JREADS

C       INTEGERS FROM TICOMN

        WRITE (JGIVEN) KTIMTI, NGIVTI, KADPTI, NMAXTI, IMPLTI,
     1                 KFACTI
C       --------------
C       ARRAY INTEGERS
C       --------------

C       INTEGERS FROM A2COMN

        WRITE (JGIVEN) (ICELA2(LC), LC = 1, NCELA2)

C       INTEGERS FROM CHCOMN

        DO 300 IR = 1, NREACH
          WRITE (JGIVEN)  NSRKCH(IR)
          WRITE (JGIVEN) (IALPCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (IBETCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (IALOCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (IBTOCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (ITABCH(IS,IR), IS = 1, NSPECH)
300     CONTINUE

C       INTEGERS FROM G2COMN

        DO 310 LC = 1, NCELG2
          WRITE (JGIVEN) (ICELG2(IP,LC), IP = 1, MCELLP), KAUXG2(LC)
310     CONTINUE

        DO 320 IB = 1, NBNDG2
          WRITE (JGIVEN) (IBNDG2(IP,IB), IP = 1, MBONDP)
320     CONTINUE

        DO 330 IN = 1, NNODG2
          WRITE (JGIVEN) (NEIBG2(IP,IN), IP = 1, MNEIBP)
330     CONTINUE

        DO 340 LV = -MLVLG2, MLVLG2
```

```
        WRITE (JGIVEN) (ILVLG2(IP,LV), IP = 1, 3)
340     CONTINUE

        WRITE (JGIVEN) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

C       ----------------------
C       NON-ARRAY REAL NUMBERS
C       ----------------------

C       REAL NUMBERS FROM A2COMN

        WRITE (JGIVEN) ALPHA2, BETAA2, GAMMA2, DELTA2, THRDA2, THRCA2

C       REAL NUMBERS FROM CHCOMN

        WRITE (JGIVEN) TREFCH, PRESCH, YNRTCH, TRIGCH

C       REAL NUMBERS FROM E2COMN

        WRITE (JGIVEN) SDELE2, SMAXE2, SMINE2, EPSLE2

C       REAL NUMBERS FROM FLCOMN

        WRITE (JGIVEN) TREFFL, PRESFL, UGASFL, AMCHFL, DISTFL,
     1                 RHORFL, UREFFL, FMREFL, WDREFL, AMWTFL,
     2                 GAMAFL

C       REAL NUMBERS FROM FRCOMN

        WRITE (JGIVEN) RHORFR, UCOMFR, VCOMFR, PRESFR, PBPIFR

C       REAL NUMBERS FROM TICOMN

        WRITE (JGIVEN) CFLNTI, TIMXTI, TIMNTI, EPS1TI, EPSOTI,
     1                 DTCNTI, FCTRTI, ERRMTI

C       WRITE THE CPU TIME HERE

        IF (MRKDA2(3) .EQ. -99) THEN
           ZCUM = WORKA2(3)
        ELSE
           CALL TIMERR (JOUTAL, ZCUM, 'PSWRTU')
        ENDIF
        WRITE (JGIVEN) ZCUM

C       ------------------
C       ARRAY REAL NUMBERS
C       ------------------

C       REAL NUMBERS FROM CHCOMN

        WRITE (JGIVEN) (PREFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (PREBCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (PREECH(IR), IR = 1, NREACH)

        WRITE (JGIVEN) (EXPFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (EXPBCH(IR), IR = 1, NREACH)
```

```
      WRITE (JGIVEN) (EXPECH(IR), IR = 1, NREACH)

      WRITE (JGIVEN) (ENEFCH(IR), IR = 1, NREACH)
      WRITE (JGIVEN) (ENEBCH(IR), IR = 1, NREACH)
      WRITE (JGIVEN) (ENEECH(IR), IR = 1, NREACH)

      WRITE (JGIVEN) (SPCPCH(IS), IS = 1, NSPECH)
      WRITE (JGIVEN) (SPCVCH(IS), IS = 1, NSPECH)
      WRITE (JGIVEN) (SPBSCH(IS), IS = 1, NSPECH)
      WRITE (JGIVEN) (FMHTCH(IS), IS = 1, NSPECH)
      WRITE (JGIVEN) (YSPECH(IS), IS = 1, NSPECH)
      WRITE (JGIVEN) (AMWTCH(IS), IS = 1, NSPECH)
      WRITE (JGIVEN) (ENTRCH(IS), IS = 1, NSPECH)

      DO 400 IR = 1, NREACH
        WRITE (JGIVEN) (BMIACH(IS,IR), IS = 1, NSPECH)
400   CONTINUE

C     REAL NUMBERS FROM E2COMN
C     WRITE (JGIVEN) (SIGGE2(IN), IN = 1, NNODG2)

C     REAL NUMBERS FROM FRCOMN

      WRITE (JGIVEN) (DPENFR(IN), IN = 1, MEQNFL)

C     REAL NUMBERS FROM G2COMN

      DO 410 IN = 1, NNODG2
         WRITE (JGIVEN) (DPENG2(IQ,IN), IQ = 1, NEQNFL)
410   CONTINUE

      DO 420 IN = 1, NNODG2
         WRITE (JGIVEN) (GEOMG2(IP,IN), IP = 1, MGEOMP)
420   CONTINUE

      WRITE (JGIVEN) (PRESG2(IN), IN = 1, NNODG2)
      WRITE (JGIVEN) (TEMPG2(IN), IN = 1, NNODG2)

C     ---------------
C     OTHER VARIABLES
C     ---------------

      PHI   = APASKY(1)
      RHOD  = APASKY(2)
      WRITE (JGIVEN) MTITLE
      WRITE (JGIVEN) PHI, RHOD

      RETURN
      END
```

PSWRT2

```
      SUBROUTINE PSWRT2 (JGIVEN)
```

```
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'H2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
      INCLUDE 'PRCOMN.INC'
      INCLUDE 'TICOMN.INC'
      INCLUDE 'TVCOMN.INC'

C***********************************************************************

C     THIS SUBROUTINE WRITES ALL THE INFORMATION ABOUT THE POINTER
C     SYSTEM AND ALL THE OTHER ARRAYS ON UNIT JGIVEN

C***********************************************************************

C     --------------
C     INITIALIZATION
C     --------------

      MCELLP = 10
      MGEOMP = 2
      MBONDP = 5
      MNEIBP = 4

      EPSOLD = EPSLE2
      IDBOLD = IDBGFR

C     SEE IF TEMPORALLY VARYING CONDITIONS WERE USED
      IF (KPERFR .EQ. 1) THEN
          PBPIFR = FLOWTV
          EPSLE2 = FREQTV
          IDBGFR = NINT(100.*AMPLTV)
      ENDIF

C     ------------------
C     NON-ARRAY INTEGERS
C     ------------------

C     WRITE ALL THE NON-ARRAY INTEGERS FIRST
C     INTEGERS FORM PARMV2

      WRITE (JGIVEN,1) NEQNFL, NREACH, NSPECH, NNODG2, NCELG2,
     1                 NBNDG2, NLVLG2, NEQBAS, KROGER
1     FORMAT(8I10)

C     INTEGERS FROM A2COMN

      WRITE (JGIVEN,1) NXTDA2, METHA2, NCELA2, K1ADA2, K2ADA2,
     1                 MTYPA2, NPLCA2, IDBGA2, MITRA2, KCHKA2,
```

```
     2                  MTHRA2, KPLTA2, KMERA2

C       INTEGERS FROM CHCOMN

        WRITE (JGIVEN,1) IDBGCH, NINRCH, NEQSCH

C       INTEGERS FROM E2COMN

        KSRTE2 = 1
        WRITE (JGIVEN,1) IDBGE2, MITRE2, KSRTE2, KONVE2, KEQNE2

C       INTEGERS FROM FLCOMN

        WRITE (JGIVEN,1) IDBGFL

C       INTEGERS FROM FRCOMN

        WRITE (JGIVEN,1) IDBGFR, KPERFR, MCYCFR, NCYCFR

C       INTEGERS FROM G2COMN

        WRITE (JGIVEN,1) IDBGG2, MALVG2, NCRSG2

C       INTEGERS FROM IOCOMN

        WRITE (JGIVEN,1) JTERMI, JTERMO, JPRINT, JCARDS, JREADI,
     1                   JREADG, JREADC, JREADD, JREADF, JOUTAL,
     2                   JHISTO, JGIVEN, JPNTWR, JDUMY1, JDUMY2,
     3                   JDUMY3, JDUMY4, JDEBUG, JREADS

C       INTEGERS FROM TICOMN

        WRITE (JGIVEN,1) KTIMTI, NGIVTI, KADPTI, NMAXTI, IMPLTI,
     1                   KFACTI
C       -------------
C       ARRAY INTEGERS
C       -------------

C       INTEGERS FROM A2COMN

        WRITE (JGIVEN,1) (ICELA2(LC), LC = 1, NCELA2)

C       INTEGERS FROM CHCOMN

        DO 300 IR = 1, NREACH
          WRITE (JGIVEN,1)  NSRKCH(IR)
          WRITE (JGIVEN,1) (IALPCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN,1) (IBETCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN,1) (IALOCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN,1) (IBTOCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN,1) (ITABCH(IS,IR), IS = 1, NSPECH)
300     CONTINUE

C       INTEGERS FROM G2COMN

        DO 310 LC = 1, NCELG2
          WRITE (JGIVEN,1) (ICELG2(IP,LC), IP = 1, MCELLP), KAUXG2(LC)
```

```
310     CONTINUE

        DO 320 IB = 1, NBNDG2
          WRITE (JGIVEN,1) (IBNDG2(IP,IB), IP = 1, MBONDP)
320     CONTINUE

        DO 330 IN = 1, NNODG2
          WRITE (JGIVEN,1) (NEIBG2(IP,IN), IP = 1, MNEIBP)
330     CONTINUE

        DO 340 LV = -MLVLG2, MLVLG2
          WRITE (JGIVEN,1) (ILVLG2(IP,LV), IP = 1, 3)
340     CONTINUE

        WRITE (JGIVEN,1) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

C       ----------------------
C       NON-ARRAY REAL NUMBERS
C       ----------------------

2       FORMAT(8E15.8)

C       REAL NUMBERS FROM A2COMN

        WRITE (JGIVEN,2) ALPHA2, BETAA2, GAMMA2, DELTA2, THRDA2, THRCA2

C       REAL NUMBERS FROM CHCOMN

        WRITE (JGIVEN,2) TREFCH, PRESCH, YNRTCH, TRIGCH

C       REAL NUMBERS FROM E2COMN

        WRITE (JGIVEN,2) SDELE2, SMAXE2, SMINE2, EPSLE2

C       REAL NUMBERS FROM FLCOMN

        WRITE (JGIVEN,2) TREFFL, PRESFL, UGASFL, AMCHFL, DISTFL,
     1                   RHORFL, UREFFL, FMREFL, WDREFL, AMWTFL,
     2                   GAMAFL

C       REAL NUMBERS FROM FRCOMN

        WRITE (JGIVEN,2) RHORFR, UCOMFR, VCOMFR, PRESFR, PBPIFR

C       REAL NUMBERS FROM TICOMN

        WRITE (JGIVEN,2) CFLNTI, TIMXTI, TIMNTI, EPS1TI, EPSOTI,
     1                   DTCNTI, FCTRTI, ERRMTI

C       WRITE THE CPU TIME HERE

        IF (MRKDA2(3) .EQ. -99) THEN
           ZCUM = WORKA2(3)
        ELSE
           CALL TIMERR (JOUTAL, ZCUM, 'PSWRT2')
        ENDIF
        WRITE (JGIVEN,2) ZCUM
```

```
C       ---------------
C       ARRAY REAL NUMBERS
C       -----------------

C       REAL NUMBERS FROM CHCOMN

        WRITE (JGIVEN,2) (PREFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN,2) (PREBCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN,2) (PREECH(IR), IR = 1, NREACH)

        WRITE (JGIVEN,2) (EXPFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN,2) (EXPBCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN,2) (EXPECH(IR), IR = 1, NREACH)

        WRITE (JGIVEN,2) (ENEFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN,2) (ENEBCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN,2) (ENEECH(IR), IR = 1, NREACH)

        WRITE (JGIVEN,2) (SPCPCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN,2) (SPCVCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN,2) (SPBSCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN,2) (FMHTCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN,2) (YSPECH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN,2) (AMWTCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN,2) (ENTRCH(IS), IS = 1, NSPECH)

        DO 400 IR = 1, NREACH
          WRITE (JGIVEN,2) (BMIACH(IS,IR), IS = 1, NSPECH)
400     CONTINUE

C       REAL NUMBERS FROM E2COMN
C       WRITE (JGIVEN,2) (SIGGE2(IN), IN = 1, NNODG2)

C       REAL NUMBERS FROM FRCOMN

        WRITE (JGIVEN,2) (DPENFR(IN), IN = 1, MEQNFL)

C       REAL NUMBERS FROM G2COMN

        DO 410 IN = 1, NNODG2
           WRITE (JGIVEN,2) (DPENG2(IQ,IN), IQ = 1, NEQNFL)
410     CONTINUE

        DO 420 IN = 1, NNODG2
           WRITE (JGIVEN,2) (GEOMG2(IP,IN), IP = 1, MGEOMP)
420     CONTINUE

        WRITE (JGIVEN,2) (PRESG2(IN), IN = 1, NNODG2)
        WRITE (JGIVEN,2) (TEMPG2(IN), IN = 1, NNODG2)

C       ---------------
C       OTHER VARIABLES
C       ---------------

3       FORMAT(A80)
        PHI   = APASKY(1)
```

```
      RHOD = APASKY(2)
      WRITE(JGIVEN,3) MTITLE
      WRITE (JGIVEN,2) PHI, RHOD

      EPSLE2 = EPSOLD
      IDBGFR = IDBOLD

      RETURN
      END
```

PSWRTU

```
      SUBROUTINE PSWRTU (JGIVEN)

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'H2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
      INCLUDE 'PRCOMN.INC'
      INCLUDE 'TICOMN.INC'
      INCLUDE 'TVCOMN.INC'

C***********************************************************************

C     THIS SUBROUTINE WRITES ALL THE INFORMATION ABOUT THE POINTER
C     SYSTEM AND ALL THE OTHER ARRAYS ON UNIT JGIVEN

C***********************************************************************

C     --------------
C     INITIALIZATION
C     --------------

      MCELLP = 10
      MGEOMP = 2
      MBONDP = 5
      MNEIBP = 4
      EPSOLD = EPSLE2
      IDBOLD = IDBGFR

C     SEE IF TEMPORALLY VARYING CONDITIONS WERE USED
      IF (KPERFR .EQ. 1) THEN
          PBPIFR = FLOWTV
          EPSLE2 = FREQTV
          IDBGFR = NINT(100.*AMPLTV)
      ENDIF
```

```
C       -----------------
C       NON-ARRAY INTEGERS
C       -----------------

C       WRITE ALL THE NON-ARRAY INTEGERS FIRST
C       INTEGERS FORM PARMV2

        WRITE (JGIVEN) NEQNFL, NREACH, NSPECH, NNODG2, NCELG2,
     1                   NBNDG2, NLVLG2, NEQBAS, KROGER

C       INTEGERS FROM A2COMN

        WRITE (JGIVEN) NXTDA2, METHA2, NCELA2, K1ADA2, K2ADA2,
     1                 MTYPA2, NPLCA2, IDBGA2, MITRA2, KCHKA2,
     2                 MTHRA2, KPLTA2, KMERA2

C       INTEGERS FROM CHCOMN

        WRITE (JGIVEN) IDBGCH, NINRCH, NEQSCH

C       INTEGERS FROM E2COMN

        KSRTE2 = 1001
        WRITE (JGIVEN) IDBGE2, MITRE2, KSRTE2, KONVE2, KEQNE2

C       INTEGERS FROM FLCOMN

        WRITE (JGIVEN) IDBGFL

C       INTEGERS FROM FRCOMN

        WRITE (JGIVEN) IDBGFR, KPERFR, MCYCFR, NCYCFR

C       INTEGERS FROM G2COMN

        WRITE (JGIVEN) IDBGG2, MALVG2, NCRSG2

C       INTEGERS FROM IOCOMN

        WRITE (JGIVEN) JTERMI, JTERMO, JPRINT, JCARDS, JREADI,
     1                 JREADG, JREADC, JREADD, JREADF, JOUTAL,
     2                 JHISTO, JGIVEN, JPNTWR, JDUMY1, JDUMY2,
     3                 JDUMY3, JDUMY4, JDEBUG, JREADS

C       INTEGERS FROM TICOMN

        WRITE (JGIVEN) KTIMTI, NGIVTI, KADPTI, NMAXTI, IMPLTI,
     1                 KFACTI
C       --------------
C       ARRAY INTEGERS
C       --------------

C       INTEGERS FROM A2COMN

        WRITE (JGIVEN) (ICELA2(LC), LC = 1, NCELA2)
```

```
C       INTEGERS FROM CHCOMN

        DO 300 IR = 1, NREACH
          WRITE (JGIVEN)  NSRKCH(IR)
          WRITE (JGIVEN) (IALPCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (IBETCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (IALOCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (IBTOCH(IS,IR), IS = 1, NSPECH)
          WRITE (JGIVEN) (ITABCH(IS,IR), IS = 1, NSPECH)
300     CONTINUE

C       INTEGERS FROM G2COMN

        DO 310 LC = 1, NCELG2
          WRITE (JGIVEN) (ICELG2(IP,LC), IP = 1, MCELLP), KAUXG2(LC)
310     CONTINUE

        DO 320 IB = 1, NBNDG2
          WRITE (JGIVEN) (IBNDG2(IP,IB), IP = 1, MBONDP)
320     CONTINUE

        DO 330 IN = 1, NNODG2
          WRITE (JGIVEN) (NEIBG2(IP,IN), IP = 1, MNEIBP)
330     CONTINUE

        DO 340 LV = -MLVLG2, MLVLG2
          WRITE (JGIVEN) (ILVLG2(IP,LV), IP = 1, 3)
340     CONTINUE

        WRITE (JGIVEN) (NBCPG2(IP,1),IP=1,4),(NBCPG2(IP,2),IP=1,4)

C       ----------------------
C       NON-ARRAY REAL NUMBERS
C       ----------------------

C       REAL NUMBERS FROM A2COMN

        WRITE (JGIVEN) ALPHA2, BETAA2, GAMMA2, DELTA2, THRDA2, THRCA2

C       REAL NUMBERS FROM CHCOMN

        WRITE (JGIVEN) TREFCH, PRESCH, YNRTCH, TRIGCH

C       REAL NUMBERS FROM E2COMN

        WRITE (JGIVEN) SDELE2, SMAXE2, SMINE2, EPSLE2

C       REAL NUMBERS FROM FLCOMN

        WRITE (JGIVEN) TREFFL, PRESFL, UGASFL, AMCHFL, DISTFL,
     1                 RHORFL, UREFFL, FMREFL, WDREFL, AMWTFL,
     2                 GAMAFL

C       REAL NUMBERS FROM FRCOMN

        WRITE (JGIVEN) RHORFR, UCOMFR, VCOMFR, PRESFR, PBPIFR
```

```
C       REAL NUMBERS FROM TICOMN

        WRITE (JGIVEN) CFLNTI, TIMXTI, TIMNTI, EPS1TI, EPSOTI,
     1                 DTCNTI, FCTRTI, ERRMTI

C       WRITE THE CPU TIME HERE

        IF (MRKDA2(3) .EQ. -99) THEN
           ZCUM = WORKA2(3)
        ELSE
           CALL TIMERR (JOUTAL, ZCUM, 'PSWRTU')
        ENDIF
        WRITE (JGIVEN) ZCUM

C       ------------------
C       ARRAY REAL NUMBERS
C       ------------------

C       REAL NUMBERS FROM CHCOMN

        WRITE (JGIVEN) (PREFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (PREBCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (PREECH(IR), IR = 1, NREACH)

        WRITE (JGIVEN) (EXPFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (EXPBCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (EXPECH(IR), IR = 1, NREACH)

        WRITE (JGIVEN) (ENEFCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (ENEBCH(IR), IR = 1, NREACH)
        WRITE (JGIVEN) (ENEECH(IR), IR = 1, NREACH)

        WRITE (JGIVEN) (SPCPCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN) (SPCVCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN) (SPBSCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN) (FMHTCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN) (YSPECH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN) (AMWTCH(IS), IS = 1, NSPECH)
        WRITE (JGIVEN) (ENTRCH(IS), IS = 1, NSPECH)

        DO 400 IR = 1, NREACH
          WRITE (JGIVEN) (BMIACH(IS,IR), IS = 1, NSPECH)
400     CONTINUE

C       REAL NUMBERS FROM E2COMN
C       WRITE (JGIVEN) (SIGGE2(IN), IN = 1, NNODG2)

C       REAL NUMBERS FROM FRCOMN

        WRITE (JGIVEN) (DPENFR(IN), IN = 1, MEQNFL)

C       REAL NUMBERS FROM G2COMN

        DO 410 IN = 1, NNODG2
           WRITE (JGIVEN) (DPENG2(IQ,IN), IQ = 1, NEQNFL)
410     CONTINUE
```

```
      DO 420 IN = 1, NNODG2
         WRITE (JGIVEN) (GEOMG2(IP,IN), IP = 1, MGEOMP)
420   CONTINUE

      WRITE (JGIVEN) (PRESG2(IN), IN = 1, NNODG2)
      WRITE (JGIVEN) (TEMPG2(IN), IN = 1, NNODG2)

C     ---------------
C     OTHER VARIABLES
C     ---------------

      PHI   = APASKY(1)
      RHOD  = APASKY(2)
      WRITE (JGIVEN) MTITLE
      WRITE (JGIVEN) PHI, RHOD

      EPSLE2 = EPSOLD
      IDBGFR = IDBOLD

      RETURN
      END
```

PTIMP2

```
      SUBROUTINE PTIMP2 (INODE, ICELL, DELN)

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'TICOMN.INC'
      COMMON/WUCOMN/ WUJACO
      DIMENSION WUJACO(MEQNFL,MEQNFL)
      DIMENSION DELS (MEQNFL), DELN (MEQNFL), SS (MEQNFL,MEQNFL)

C*********************************************************************

C     THIS SUBROUTINE APPLIES THE POINT-IMPLICIT APPROACH TO THE NI
C     SCHEME AT THE GIVEN NODE INODE DUE TO THE CELL ICELL

C*********************************************************************

C     SET UP THE PRECONDITIONING MATRIX

      DO 20 J = 1, NEQNFL
         DO 10 K = 1, NEQNFL
            SS(J,K) = -WUJACO(J,K)*CELLTI(ICELL)
10       CONTINUE
         SS(J,J) = 1. + SS(J,J)
         DELS(J) = DELN(J)
20    CONTINUE

C     NOW INVERT THE PRECONDITIONING MATRIX
```

```
      CALL GAUSS3 (SS, DELN, DELS, NEQNFL, MEQNFL)

C     NOW RESET THE CHANGE VARIABLE

      DO 60 J = 1, NEQNFL
         DELN(J) = DELS(J)
60    CONTINUE

      RETURN
      END
```

ROGERC

```
      SUBROUTINE ROGERC
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'PRCOMN.INC'
C
C*********************************************************************

C     THIS SUBROUTINE CHANGES THE ROGERS AND CHINITZ MODEL FROM FOUR
C     SPECIES TO THREE SPECIES COMPUTATIONS AND VICE VERSA.

C*********************************************************************
C
      IF (KROGER .NE. 1) RETURN

      IF (NINRCH .GT. 0) THEN
         WRITE(6,*) ' CHANGING FROM 3 TO 4 SPECIES'
         DO 200 INODE = 1, NNODG2

C           COMPUTE THE MASS FRACTIONS FOR EACH SPECIES
            SUMY   = 0.
            YUPPER = 1. - YNRTCH
            RHORPR = DPENG2(1,INODE)

            DO 190 IS = 1, NEQSCH
               JS         = NEQBAS + IS
               YSPEPR(IS) = DPENG2(JS,INODE)/RHORPR
               IF (YSPEPR(IS) .LT. 0.) THEN
                 YSPEPR(IS)       = 0.
                 DPENG2(JS,INODE) = 0.
               ENDIF
               IF (YSPEPR(IS) .GT. 1.) THEN
                 YSPEPR(IS)       = 1.
                 DPENG2(JS,INODE) = YUPPER*RHORPR
               ENDIF
```

```
                SUMY       = SUMY + YSPEPR(IS)
190           CONTINUE
C             - THE FOLLOWING IS FOR SPECIES 4 = NEQSCH+1
              YSPEPR(NEQSCH+1) = 1. - SUMY - YNRTCH
              IF (YSPEPR(NEQSCH+1) .LT. 0.) YSPEPR(NEQSCH+1) = 0.
C             ADJUST THE NEWLY DEFINED VARIABLE AT THIS NODE
              DPENG2(NEQNFL+1,INODE) = RHORPR*YSPEPR(NEQSCH+1)
200         CONTINUE

C           NOW ADJUST THE NUMBER OF EQUATIONS
            YNRTCH = 0.
            NEQNFL = NEQNFL + 1
            NEQSCH = NEQSCH + 1
            NINRCH = NINRCH - 1
C
          ELSE
C
            WRITE(6,*) ' CHANGING FROM 4 TO 3 SPECIES'

C           NOW ADJUST THE NUMBER OF EQUATIONS
            YNRTCH = YSPECH(5)
            NEQNFL = NEQNFL - 1
            NEQSCH = NEQSCH - 1
            NINRCH = NINRCH + 1

          ENDIF

          RETURN
          END
```

SETUPU

```
          SUBROUTINE SETUPU
C
          INCLUDE 'PRECIS.INC'
          INCLUDE 'IOCOMN.INC'
C
C*****************************************************************
C
C         THIS SUBROUTINE DOES ALL THE INPUT/OUTPUT UNIT INITIALIZATIONS.
C
C*****************************************************************
C
C         JTERMI = TERMINAL INPUT
C         JTERMO = TERMINAL OUTPUT
C         JPRINT = PRINT UNIT
C         JCARDS = CARD READER
C         JREADS = FILE CONTAINING THE SCHEDULE PROGRAM
C
C         JHISTO = HISTORY FILE -- STATISTICAL DATA FOR EACH ITERATION
C         JOUTAL = OUTPUT FILE  -- CONTAINS ALL THE OUTPUT
C         JREADI = INPUTI.DAT   -- CONTAINS INPUT RECORDS
C         JREADF = INPUTF.DAT   -- CONTAINS OUTLET CONDITIONS
```

```
C       JREADG = INPUTG.DAT   -- CONTAINS GEOMETRIC INFORMATION
C       JREADC = INPUTC.DAT   -- CONTAINS CHEMISTRY VARIABLES
C       JREADD = INPUTD.DAT   -- CONTAINS INITIAL DPENDENT VARIABLES
C       JDUMYN = DUMMY UNITS (N = 1,2,3,4)
C       JDEBUG = DEBUG UNIT
C       JPOINT = POINT.DAT    -- CONTAINS ALL THE POINTER INFORMATION FOR
C                                RESTART PURPOSES
C
C       SET THE INPUT/OUTPUT UNIT NUMBERS
C
        JTERMI = 5
        JTERMO = 6
        JPRINT = 7
        JCARDS = 8
C
        JREADI = 11
        JREADG = 12
        JREADC = 13
        JREADD = 14
        JPNTRE = 15
        JREADF = 16
C
        JOUTAL = 21
        JHISTO = 22
        JPNTWR = 23
        JDEBUG = 24
C
        JDUMY1 = 31
        JDUMY2 = 32
        JDUMY3 = 33
        JDUMY4 = 34
C
C       INITIALIZE TIMER
C
        CALL TIMERR(JOUTAL, ZCUM, ' ')
C
        RETURN
        END
```

SHORTG

```
        SUBROUTINE SHORTG(IPATH)
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        DIMENSION VERTEX(2,10)
C
C*****************************************************************
C
C       THIS SUBROUTINE DETERMINES A SHORTER GRID DOMAIN FOR INTEGRATION
```

```
C       PURPOSES, SO THAT THE CONVERGENCE TO STEADY STATE CAN BE
C       HASTENED
C
C*********************************************************************

        JPOLYM = 85
        JBIGER = 86
        JSMALL = 87

        GOTO (1000,2000,3000), IPATH
        RETURN
C
C       INITIALIZE THE GRID
C
1000    CONTINUE
C
C       FREEZE THE ADAPTIVE PROCEDURE
C
        METHA2 = 0
        JREADS = 0
C
        OPEN (UNIT=JPOLYM,FILE='INPUTK.DAT',FORM='FORMATTED' ,
     1       STATUS='OLD')
        OPEN (UNIT=JBIGER,FILE='JBIGER.DAT',FORM='FORMATTED',
     1       STATUS='NEW')
        OPEN (UNIT=JSMALL,FILE='JSMALL.DAT',FORM='FORMATTED',
     1       STATUS='NEW')
C
C       THE CURTAILED GRID IS READ AS A POLYGONAL REGION
C       READ THE TOTAL NUMBER OF VERTICES IN THE POLYGON
        READ (JPOLYM,*) NVERT
        IF (NVERT .LT. 3 .OR. NVERT .GT. 10) RETURN

C       READ THE TYPE OF BOUNDARY CONDITION FOR THE LEFT-MOST BOUNDARY
C       OF THE BLOCK UNDER CONSIDERATION
        READ (JPOLYM,*) IBCTYP

        DO 1010 IVERT = 1, NVERT
           READ (JPOLYM,*) VERTEX(1,IVERT), VERTEX(2,IVERT)
1010    CONTINUE

        CLOSE (JPLOYM)
        NPLCA2 = NCELA2
        NBNDPV = NBNDG2
        NCELA2 = 0
C
C       SAVE THE OLD POINTERS

        WRITE(JBIGER,667) NPLCA2, NBNDPV
        WRITE(JBIGER,667) (ICELA2(JCELL),JCELL=1,NPLCA2)
667     FORMAT(15I7)

        DO 1020 JCELL = 1, NPLCA2
C
C          FIND THE ACTUAL CELL AND ITS CENTER
           ICELL = ICELA2(JCELL)
           KSW   = ICELG2(2,ICELL)
```

```
            KSE   = ICELG2(4,ICELL)
            KNE   = ICELG2(6,ICELL)
            KNW   = ICELG2(8,ICELL)
            XSW   = GEOMG2(1,KSW)
            XSE   = GEOMG2(1,KSE)
            XNE   = GEOMG2(1,KNE)
            XNW   = GEOMG2(1,KNW)
            YSW   = GEOMG2(2,KSW)
            YSE   = GEOMG2(2,KSE)
            YNE   = GEOMG2(2,KNE)
            YNW   = GEOMG2(2,KNW)
            XC    = 0.25*(XSW+XSE+XNE+XNW)
            YC    = 0.25*(YSW+YSE+YNE+YNW)
C
C           DETERMINE IF THIS CELL IS INSIDE THE POLYGON
            CALL INSIDE (IN, VERTEX, NVERT, XC, YC)

            IF (IN .EQ. 1) THEN
                  NCELA2 = NCELA2 + 1
               ICELA2(NCELA2) = ICELL
            ENDIF

1020     CONTINUE
C
C        SET THE INLET BOUNDARY
C
         IF (IBCTYP .EQ. 2) THEN
            DO 1030 JCELL = 1, NCELA2
C
C              FIND THE CELL AND ITS TWO LEFT NODES

               ICELL = ICELA2(JCELL)
               KSW   = ICELG2(2,ICELL)
               KNW   = ICELG2(8,ICELL)

C              FIND THE TWO NEIGHBOR CELLS ON THE LEFT

               NB1 = NEIBG2(4,KSW)
               NB2 = NEIBG2(1,KNW)

C              IF THESE NODES ARE NOT ON A BOUNDARY, AND IF SO CHECK THE
C              NODES ON THE LEFT, THEY SHOULD NOT BE IN THE BLOCK ITSELF

               IF (NB1 .NE. 0) THEN

                  DO 1021 KCELL = 1, NCELA2
                     IF (NB1 .EQ. ICELA2(KCELL)) GOTO 1022
1021              CONTINUE
C
C                 THE CELL IS ON A BOUNDARY, NOW CHECK THE NODE ITSELF
                  DO 911 IBND = 1, NBNDG2
                      IF (KSW .EQ. IBNDG2(1,IBND)) GOTO 1022
911               CONTINUE

C                 ITS A BOUNADRY POINT AND NOT ALREADY MARKED
                  NBNDG2 = NBNDG2 + 1
                  IBNDG2(1,NBNDG2) = KSW
```

```
                IBNDG2(5,NBNDG2) = 2

              ENDIF

1022          IF (NB2 .NE. 0) THEN

                DO 1023 KCELL = 1, NCELA2
                   IF (NB2 .EQ. ICELA2(KCELL)) GOTO 1024
1023            CONTINUE
C
C               THE CELL IS ON A BOUNDARY, NOW CHECK THE NODE ITSELF
                DO 913 IBND = 1, NBNDG2
                    IF (KSE .EQ. IBNDG2(1,IBND)) GOTO 1024
913             CONTINUE

C               ITS A BOUNADRY POINT AND NOT ALREADY MARKED
                NBNDG2 = NBNDG2 + 1
                IBNDG2(1,NBNDG2) = KSE
                IBNDG2(5,NBNDG2) = 2

              ENDIF

1024          CONTINUE

1030       CONTINUE

        ENDIF
C
C       SAVE THE CHANGED POINTERS
C
        WRITE(JSMALL,667) NCELA2, NBNDG2
        WRITE(JSMALL,667) (ICELA2(JCELL),JCELL=1,NCELA2)

        CLOSE (JSMALL)
        CLOSE (JBIGER)

        RETURN

2000    CONTINUE
C
C       READ THE FULL DOMAIN AGAIN
C
        OPEN (UNIT=JBIGER,FILE='JBIGER.DAT',FORM='FORMATTED',
     1        STATUS='OLD')
        READ (JBIGER,667) NCELA2, NBNDG2
        READ (JBIGER,667) (ICELA2(JCELL),JCELL=1,NCELA2)

        CLOSE (JBIGER)

        RETURN

3000    CONTINUE
C
C       READ THE CURTAILED DOMAIN AGAIN
C
        OPEN (UNIT=JSMALL,FILE='JSMALL.DAT',FORM='FORMATTED',
     1        STATUS='OLD')
```

```
      READ (JSMALL,667) NCELA2, NBNDG2
      READ (JSMALL,667) (ICELA2(JCELL),JCELL=1,NCELA2)

      CLOSE (JSMALL)

      RETURN
      END
```

TIINI2

```
      SUBROUTINE TIINI2
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'TICOMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
C
C*********************************************************************
C
C     THIS SUBROUTINE INITIALIZES ALL THE COMMON BLOCK ARRAYS THAT
C     ARE TO BE USED FOR TEMPORAL EMBEDDING
C
C*********************************************************************
C
C     SET THE CORANT NUMBER
      CFLNTI = APASKY( 3)
C
C     SET THE CONSTANT CELL TIME STEP; NEGATIVE VALUE MEANS THAT
C     A LOCAL VALUE WILL BE COMPUTED
      DTCNTI = APASKY(35)
C
C     SET THE EPSILON VALUE FOR TEMPORAL EMBEDDING
      EPS1TI = APASKY(22)
      EPSOTI = APASKY(23)

C     MAXIMUM ERROR ABOVE WHICH EPS1TI WILL BE DECREASED
C     MAXIMUM ERROR USED IN DETERMINING THE TEMPORAL CELL FACTOR
      ERRMTI = APASKY(27)
C
C     INITIALIZE THE FACTOR FOR ADJUSTING CELL TIME STEPS
      FCTRTI = APASKY(34)
C
C     SET THE DEBUG PARAMETER FOR TI ROUTINES
      IDBGTI = IPASKY(34)

C     SET THE PARAMETER INDICATING WHETHER EXPLICIT OR IMPLICIT
C     SOURCE TERMS ARE TO BE USED; 1:EXPLICIT
      IMPLTI = IPASKY(33)
      IF (IMPLTI .NE. 1) IMPLTI = 0

C     SET UP THE CRITERION VARIABLE TO BE USED FOR TEMPORAL RESOLUTION
      KADPTI = IPASKY(24)
```

```
C       SET UP THE PARAMETER INDICATING IF RESULTS AT VARIOUS TIME LEVELS
C       ARE NEEDED
        KTIMTI = IPASKY(10)

C       SET UP THE MAXIMUM GIVEN (TEMPORAL) LEVEL OF CELLS
        NGIVTI = IPASKY(7)

C       SET THE NUMBER OF CELLS TO BE MOVED AWAY FROM THE NODIT'S
        KFACTI = IPASKY(17)

C       SEE IF THE DIFFERENCE OF SPECIES MASS FRACTIONS IS TO BE USED
C       FOR LIMITING THE TIME-STEPS
        KDIFTI = IPASKY(39)

C       SET THE MAXIMUM AND MINIMUM TIMES OF THE RUN
        TIMXTI = APASKY(20)
        TIMNTI = APASKY(24)

C       SET THE MAXIMUM CFL NUMBER
        CFLXTI = APASKY(41)
C
C       PRINT OUT PARAMETERS
C
        IF (IDBGTI .NE. 1 .AND. IDBGTI .LT. 1000)  RETURN
        WRITE(JDEBUG,1000)
        WRITE(JDEBUG,1100)
        WRITE(JDEBUG,1200)
        WRITE(JDEBUG,1300) CFLNTI, EPSOTI, EPS1TI, TIMXTI, TIMNTI,
     1                     ERRMTI, FCTRTI, DTCNTI,
     2                     KADPTI, IMPLTI, KTIMTI, NGIVTI, KFACTI

C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT(//10X,'----------------------' )
1100    FORMAT(  10X,'DEBUG PRINT FROM TIINI2' )
1200    FORMAT(  10X,'----------------------'/)
1300    FORMAT( 5X, 'CFLNTI = ', G14.5, 10X, 'EPSOTI = ', G14.5/
     1          5X, 'EPS1TI = ', G14.5, 10X, 'TIMXTI = ', G14.5/
     2          5X, 'TIMNTI = ', G14.5, 10X, 'ERRMTI = ', G14.5/
     3          5X, 'FCTRTI = ', G14.5, 10X, 'DTCNTI = ', G14.5/
     4          5X, 'KADPTI = ', I5,    19X, 'IMPLTI = ', I5   /
     5          5X, 'KTIMTI = ', I5,    19X, 'NGIVTI = ', I5   /
     6          5X, 'KFACTI = ', I5                             )

        RETURN
        END
```

TIPRN2

```
        SUBROUTINE TIPRN2 (IUNIT)
```

```
      INCLUDE '[.INC] PRECIS.INC/LIST'
      INCLUDE '[.INC] PARMV2.INC/LIST'
      INCLUDE '[.INC] G2COMN.INC/LIST'
      INCLUDE '[.INC] IOCOMN.INC/LIST'
      INCLUDE '[.INC] TICOMN.INC/LIST'

C*********************************************************************
C
C     THIS SUBROUTINE PRINTS ALL TEMPORAL POINTER ARRAYS ON IUNIT
C
C*********************************************************************

      CALL HEADER(IUNIT, 'TEMPORAL CELL VARIABLES', MTITLE)

      ICLAST = ILVLTI(2,NMAXTI)
      WRITE(IUNIT, 1000) ICLAST, NMAXTI, DTMNTI
      WRITE(IUNIT, 1100)

      DO 10 JCELL=1, ICLAST
          ICELL = ICELTI(JCELL)
          NODESW = ICELG2(2,ICELL)
          NODESE = ICELG2(4,ICELL)
          NODENE = ICELG2(6,ICELL)
          NODENW = ICELG2(8,ICELL)
          WRITE(IUNIT,1200) JCELL,ICELL,NODESW,NODESE,NODENE,NODENW,
     1          GEOMG2(1,NODESW), GEOMG2(1,NODESE), GEOMG2(1,NODENE),
     2          GEOMG2(1,NODENW), CELLTI(ICELL)
10    CONTINUE

C     TEMPORAL-GRID-LEVEL ARRAY

      WRITE(IUNIT,1300)

      DO 20 ITGL = 0, NMAXTI
          NOCELL = ILVLTI(2,ITGL) - ILVLTI(1,ITGL) + 1
          WRITE(IUNIT,1400) ITGL,(ILVLTI(K,ITGL),K=1,2),NOCELL
20    CONTINUE
C
C     ----------------
C     FORMAT STATEMENTS
C     ----------------
C
1000  FORMAT (1X,'ICLAST',2X,'NMAXTI',5X,'DTMNTI'/2I7,G14.5)
1100  FORMAT(1X,'JCELL',2X,'ICELL',3X,'N-SW',2X,'N-SE',2X,'N-NE',2X,
     1          'N-NW',5X,'X-DIS-SW',6X,'X-DIS-SE',6X,'X-DIS-NE',6X,
     2          'X-DIS-NW',6X, 'CELLTI')
1200  FORMAT(1X,6(I5,1X),3X,5G14.5)
1300  FORMAT(//' TEMPORAL-GRID-LEVEL INFORMATION:'//
     1       5X,'LEVEL',4X,'START',6X,'END',5X,'# CELLS')
1400  FORMAT(1X,4(I7,3X))

      RETURN
      END
```

TVINI0

```
      SUBROUTINE TVINI0

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'CHCOMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FLCOMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'H2COMN.INC'
      INCLUDE 'HEXCOD.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'TICOMN.INC'
      INCLUDE 'TVCOMN.INC'
C
C*********************************************************************
C
C       THIS SUBROUTINE ADDS EMBEDDED CELLS ACROSS THE TEMPORALLY
C       VARYING PLANE.  THESE CELLS ARE PERMANENTLY DIVIDED AND NEVER
C       ALLOWED COLLAPSE AGAIN.  THE LEVELS OF EMBEDDING ACROSS THE
C       INLET PLANE EQUALS THE CURRENT MAXIMUM EMBEDDING LEVEL
C       THE SUBROUTINE ALSO INITIALIZES THE PERIODIC BOUNDARY CONDITIONS
C       CONSTANTS
C
C*********************************************************************
C
C       THE BASE NODE AT THE INLET PLANE IS ASSUMED TO BE 1
C
      IBASEN = 1
      IF (IADDH2 .NE. 3) RETURN
      KPERFR = 1
C
C       SET INITIAL TIME EQUAL TO ZERO
C
      TIMNTI = 0.
C
C       READ THE MEAN MASS FLOW RATE, FREQUENCY AND PERCENTAGE CHANGE
C
C       READ (JREADS,*) FLOWTV
      READ (JREADS,*) FREQTV
      READ (JREADS,*) IPERCN
C
C       INITIALIZE THE VALUES
C
      FLOWTV = DPENG2(2,IBASEN)
      AMPLTV = FLOAT(IPERCN)/100.
      WRITE(6,*) ' tvini0 FLOWTV =',FLOWTV
      WRITE(6,*) ' tvini0 FREQTV =',FREQTV
      WRITE(6,*) ' tvini0 ampltv =',amplTV
      WRITE(6,*)
      WRITE(6,*) ' THE FOLLOWING NODES ARE NOTED AS INLET'

      PBPIFR = FLOWTV
      EPSLE2 = FREQTV
```

```
      IDBGFR = IPERCN
C          -
C     NOW COMPUTE AND CORRECT THE PRIMITIVE VARIABLES
C
      RHORPR = DPENG2(1,IBASEN)
      UMASMX = FLOWTV*(1. + AMPLTV)
      UCOMPR = UMASMX/RHORPR
      VCOMPR = DPENG2(3,IBASEN)/RHORPR
      BEPSPR = DPENG2(4,IBASEN)
      BEU    = BEPSPR/RHORPR
      VEL02U = UCOMPR*UCOMPR + VCOMPR*VCOMPR
C
C     COMPUTE THE DIMENSIONAL QUANTITIES
C
      BE     = FMREFL*BEU
      VEL02  = FMREFL*VEL02U
C
      SYSHFS = 0.
      SYSCPS = 0.
      SYSBMS = 0.
      BIGAM  = 0.
C
      DO 80 IS = 1, NSPECH
          SYSHFS = SYSHFS + YSPECH(IS)*FMHTCH(IS)
          SYSCPS = SYSCPS + YSPECH(IS)*SPCPCH(IS)
          SYSBMS = SYSBMS + YSPECH(IS)*RAMWCH(IS)
          BIGAM  = BIGAM  + YSPECH(IS)*SPBSCH(IS)
80    CONTINUE

      BIGBM   = SYSCPS - UGASFL*SYSBMS
      BIGCM   = BE - 0.5*VEL02 - SYSHFS + TREFCH*SYSCPS
     1                        + 0.5*TREFCH*TREFCH*BIGAM

      IF (BIGCM .LT. 1.E-10) THEN
            WRITE(6,*) ' VELOCITY DEFECT IS TOO HIGH'
            WRITE(6,*) ' IT WILL CAUSE TEMPERATURE TO GO NEGATIVE'
      ENDIF
C
C
C
C     DIVIDE THE CELLS AT THE INLET AND PERMANENTLY MARK THEM
C
      DO 180 ILEVEL = 1, MALVG2

        INODE  = IBASEN
170     NBCELL       = NEIBG2(3,INODE)
        IF (NBCELL .EQ. 0 ) GOTO 180
        KAUXG2(NBCELL) = IOR(KAUXG2(NBCELL),KL2000)
        IWARN          = 0
        CALL  G2DIVO (NBCELL,IWARN)
        INODE          = ICELG2(8,NBCELL)
        GO TO 170

180   CONTINUE
C
C     SAVE THE NODE WHERE VALUES ARE CHANGING
```

```
      INODE  = IBASEN
      NUMNTV = 0
190   NBCELL       = NEIBG2(3,INODE)
      NUMNTV  = NUMNTV + 1
      WRITE(6,*) NUMNTV, INODE
      NODETV(NUMNTV) = INODE
      IF (NBCELL .EQ. 0 ) GOTO 200
      INODE        = ICELG2(8,NBCELL)
      GO TO 190

200   CONTINUE

      CALL A2CEWC

      RETURN
      END
```

TVINI1

```
      SUBROUTINE TVINI1

      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'HEXCOD.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'TVCOMN.INC'
C
C*********************************************************************
C
C     THE SUBROUTINE ALSO REINITIALIZES THE PERIODIC BOUNDARY CONDITIONS
C     CONSTANTS
C
C*********************************************************************
C
C     THE BASE NODE AT THE INLET PLANE IS ASSUMED TO BE 1
C
      IBASEN = 1
      IF (KPERFR .NE. 1) RETURN
C
C     INITIALIZE THE VALUES
C
C     SAVE THE NODE WHERE VALUES ARE CHANGING

      INODE  = IBASEN
      NUMNTV = 0
190   NBCELL       = NEIBG2(3,INODE)
      NUMNTV  = NUMNTV + 1
      NODETV(NUMNTV) = INODE
      INODE        = ICELG2(8,NBCELL)
      IF (NBCELL .EQ. 0 ) GOTO 200
      GO TO 190
```

```
200     CONTINUE

        RETURN
        END
```

TWOD0U

```
        PROGRAM TWODOU
C
        INCLUDE 'PRECIS.INC'
        INCLUDE 'PARMV2.INC'
        INCLUDE 'A2COMN.INC'
        INCLUDE 'E2COMN.INC'
        INCLUDE 'FRCOMN.INC'
        INCLUDE 'G2COMN.INC'
        INCLUDE 'IOCOMN.INC'
        INCLUDE 'KYCOMN.INC'
        INCLUDE 'TICOMN.INC'
        INCLUDE 'TVCOMN.INC'
C
C*********************************************************************
C
C       THIS IS THE MAIN CONTROLLING ROUTINE FOR NI'S TECHNIQUE FOR
C       SOLVING TWO-DIMENSIONAL EULER'S EQUATION INVOLVING CHEMICAL
C       REACTIONS.  IN THIS PROGRAM UNIT WE ARE ONLY INTERESTED IN
C       SOLVING UNSTEADY FLOW PROBLEMS AND HENCE TEMPORAL ADAPTATION
C       IS USED. SPATIALLY EMBEDDED MESHES CAN BE HANDLED BY USING
C       A NODE/CELL POINTER SYSTEM SIMILAR TO THAT OF BILL USAB OR
C       JOHN DANNENHOFFER.  A NEW POINTER SYSTEM IS NEEDED FOR THE
C       TEMPORAL ADAPTATION.  A THIRD POINTER SYSTEM IS USED FOR
C       CHEMICAL SPECIES AND REACTIONS.
C
C       THIS PROGRAM MUST BE USED IF A SHORTER GRID THAN WHAT POINTER
C       SYSTEM ALLOWS IS NOT TO BE USED.   FURTHERMORE IT MUST BE USED
C       WHEN TEMPORALLY VARYING INFLOW BOUNDARY CONDITIONS ARE NOT
C       TO BE UTILIZED.
C
C*********************************************************************
C
C
C       INITIALIZE ALL THE ARRAYS
C
        CALL   E2INIO

C       SET UP THE INITAL TIME FOR THIS CASE
C
        TIME   = TIMNTI
        IF (KADPTI .EQ. 99) TIME = 0.
        NTERE  = 0
        NTERT  = 0
        NTERA  = 0
        NITRE2 = 0
```

```
C       NCYCFR = 1
C
C       ----------------
C       OPTION PARAMETERS
C       ----------------
C
C       SET MAXIMUM NUMBER OF TIMES BEFORE POINTER SYSTEM IS SAVED
        MITRPS = IPASKY(27)

C       SET MAXIMUM NUMBER OF NGIVTI
        KHAFEZ = IPASKY(28)

C       MAXIMUM NUMBER OF ITERATIONS AFTER WHICH EPS1TI IS DECREASED
        MITEPS = IPASKY(32)

C       MINIMUM ERROR BELOW WHICH EPS1TI WILL BE INCREASED
        ERRMIN = APASKY(26)

C       MAXIMUM ERROR ABOVE WHICH EPS1TI WILL BE DECREASED
        ERRMTI = APASKY(27)

C       MINIMUM ALLOWABLE VALUE OF EPS1TI
        EPS1MN = APASKY(28)

C       MAXIMUM ALLOWABLE VALUE OF EPS1TI
        EPS1MX = APASKY(29)

C
C       ----------------------
C       NORMAL RUN STARTS HERE
C       ----------------------
C
C       SEE IF THE SCHEDULE PROGRAM IS NEEDED
10      CALL   E2SCHO

C       SET THE DIFFUSION COEFFICIENTS AT ALL NODES FOR FIRST TIME
        CALL E2DIFF

20      NTERE  = NTERE  + 1
        NTERT  = NTERT  + 1
        NITRE2 = NITRE2 + 1

C       SEE IF THE POINTER SYSTEM IS TO BE SAVED; IF SO ONLY UNFORMATTED
C       OUTPUT WILL BE WRIITEN

        IF (NTERE .GE. MITRPS) THEN
          NTERE  = 0
          TDUM   = TIMNTI
          TIMNTI = TIME
          WRITE(JTERMO,*) ' WRITTING UNFORMATTED OUTPUT ON',JDUMY4
C         READ THE FULL DOMAIN AGAIN IF A CURTAILED DOMAIN WAS USED
          CALL PSWRTU (JDUMY4)
          CLOSE (JDUMY4)
          TIMNTI = TDUM
          JDUMY4 = JDUMY4 + 1
          IF (JDUMY4 .EQ. 40) JDUMY4 = 34
        ENDIF
```

```
C
C       SEE IF SPATIAL ADAPTATION IS NEEDED
C       MITRA2 DENOTES THE NUMBER OF ITERATION (OR PASSES) AFTER
C       WHICH ADAPTATION IS DONE; ADAPTATION LOOP IS BY-PASSED
C       IF METHA2 = 0 (METHOD OF ADAPTATION)

        IF (METHA2 .NE. 0) THEN
           NTERA = NTERA + 1
           IF (NTERA  .EQ. 1     ) CALL  A2MTHO
           IF (NTERA  .GE. MITRA2) NTERA = 0
        ENDIF

C       SET ALL CHANGES TO ZERO FOR ALL THE NODES
        CALL E2ZERO

C       COMPUTE THE TIME STEPS FOR EACH CEWIC CELL
        CALL E2TIMO
        FCTRTI = 1.

C       COMPUTE THE NUMBER OF INTEGRATION PASSES
        KMAX   = 2**NMAXTI
        IPASSM = 2*KMAX - 1

        IF (KADPTI .EQ. 99) DTMNTI = 0.
C       COMPUTE THE CURRENT TIME OF THE RUN
C       IF UNSTEADY INLET BOUNDARY CONDITIONS ARE NOT USED THEN
C       UNCOMMENT THE FOLLOWING LINE AND COMMENT THE CONDITIONAL
C       STATEMENT INSIDE THE LOOP ITSELF FOR EFFICIENT CALCULATION
        TIME  = TIME + KMAX*DTMNTI

C       DETERMINE THE NODES AT ALL TEMPORAL LEVELS
        CALL G2HANG

        DO 30 IPASS = 1, IPASSM
C          DETERMINE THE TEMPORAL LEVEL OF CELLS TO BE INTEGRATED
           ITGL = ITLEVL(IPASS,NMAXTI)

C          CALCULATE CHANGE AND DISTRIBUTE FOR ALL CELLS ON THIS LEVEL
           CALL E2SOLO (ITGL)

C          APPLY BOUNDARY CONDITIONS
           CALL E2BCNO (ITGL)

C          COLLECT THE CONVERGENCE HISTORY IF NEED BE
           CALL E2CONO (TIME, ITGL, IPASS, IPASSM)

C          UPDATE ALL NODES AT THIS LEVEL
           CALL E2UPDO (ITGL)

C          LOOP BACK FOR NEXT TEMPORAL LEVEL CELLS

30      CONTINUE
C
C       DETERMINE ARTIFICIAL VISCOSITY COEFFICIENT AT EACH NODE AFTER
C       EACH TIME-STRIDE
C
        CALL E2DIFF
```

```
C       SEE IF TEMPORAL ADJUSTMENTS ARE NEEDED
C       IF (MITRPS .NE. 0) THEN
C         IF (ERORE2 .LT. ERRMIN) THEN
C           NITEPS = NITEPS + 1
C           IF (NITEPS .GT. MITEPS) THEN
C              EPS1NW = 1.05*EPS1TI
C              EPS1TI = MIN (EPS1NW, EPS1MX)
C              NITEPS  = 0
C              IF (NGIVTI .LT. KHAFEZ) NGIVTI = NGIVTI + 1
C           ENDIF
C         ELSE
C           NITEPS = 0
C         ENDIF
CC
C         IF (ERORE2 .GT. ERRMTI) THEN
C           EPS1NW = 0.95*EPS1TI
C           EPS1TI = MAX (EPS1NW, EPS1TI)
C           WRITE(6,*) ' EPS1TI = ',EPS1TI
C         ENDIF
C       ENDIF

        IF (NITRE2 .GE. MITRE2) GOTO 40
        IF (ERORE2 .LE. EPSLE2) GOTO 40
        IF (TIME   .GE. TIMXTI) GOTO 40
C       IF (NCYCFR .GE. MCYCFR+1) GOTO 40
        GO TO 20

40      CONTINUE

C       PRINT OUT PARAMETERS
C       IF (IDBGFL .EQ. 7 .OR. IDBGFL .GT. 1000)  THEN
C           WRITE(JDEBUG,1000)
C           WRITE(JDEBUG,1100)
C           WRITE(JDEBUG,1200)
C           WRITE(JDEBUG,1300) MITRPS, MITEPS, KTIMTI, MITRA2, NITRE2,
C     1                        KHAFEZ
C           WRITE(JDEBUG,1400) ERRMIN, ERRMTI, EPS1MN, EPS1MX, ERORE2,
C     1                        TIME  , EPS1TI
C       ENDIF
C1000   FORMAT(//10X,'----------------------' )
C1100   FORMAT(  10X,'DEBUG PRINT FROM TWODOU' )
C1200   FORMAT(  10X,'----------------------'/)
C1300   FORMAT(5X,'MITRPS = ',I5,10X,'MITEPS = ',I5/
C     1        5X,'KTIMTI = ',I5,10X,'MITRA2 = ',I5/
C     2        5X,'NITRE2 = ',I5,10X,'KHAFEZ = ',I5/)
C1400   FORMAT(5X,'ERRMIN = ',G14.5,10X,'ERRMTI = ',G14.5/
C     1        5X,'EPS1MN = ',G14.5,10X,'EPS1MX = ',G14.5/
C     2        5X,'ERORE2 = ',G14.5,10X,'TIME   = ',G14.5/
C     3        5X,'EPS1TI = ',G14.5,10X,                /)

        TIMNTI = TIME
        IF (JREADS .EQ. 0) THEN
           CALL E2FINI
           STOP ' THE END'
        ENDIF
```

```
      GOTO 10

      END -
```

TWODBC

```
      PROGRAM TWODOU
C             TWODBC.FOR
C
      INCLUDE 'PRECIS.INC'
      INCLUDE 'PARMV2.INC'
      INCLUDE 'A2COMN.INC'
      INCLUDE 'E2COMN.INC'
      INCLUDE 'FRCOMN.INC'
      INCLUDE 'G2COMN.INC'
      INCLUDE 'IOCOMN.INC'
      INCLUDE 'KYCOMN.INC'
      INCLUDE 'TICOMN.INC'
      INCLUDE 'TVCOMN.INC'
C
C*********************************************************************
C
C     THIS IS THE MAIN CONTROLLING ROUTINE FOR NI'S TECHNIQUE FOR
C     SOLVING TWO-DIMENSIONAL EULER'S EQUATION INVOLVING CHEMICAL
C     REACTIONS.  IN THIS PROGRAM UNIT WE ARE ONLY INTERESTED IN
C     SOLVING UNSTEADY FLOW PROBLEMS AND HENCE TEMPORAL ADAPTATION
C     IS USED. SPATIALLY EMBEDDED MESHES CAN BE HANDLED BY USING
C     A NODE/CELL POINTER SYSTEM SIMILAR TO THAT OF BILL USAB OR
C     JOHN DANNENHOFFER.  A NEW POINTER SYSTEM IS NEEDED FOR THE
C     TEMPORAL ADAPTATION.  A THIRD POINTER SYSTEM IS USED FOR
C     CHEMICAL SPECIES AND REACTIONS.
C
C     USE THIS PROGRAM ROUTINE IF TEMPORALLY VARYING BOUNDARY CONDTIONS
C     ARE TO BE USED
C
C*********************************************************************
C
C
C     INITIALIZE ALL THE ARRAYS
C
      CALL   E2INIO

C     SET UP THE INITAL TIME FOR THIS CASE
C
      TIME   = TIMNTI
      IF (KADPTI .EQ. 99) TIME = 0.
      NTERE  = 0
      NTERT  = 0
      NTERD  = 0
      NTERA  = 0
      NITRE2 = 0
C     NCYCFR = 1
C
```

```
C       -----------------
C       OPTION PARAMETERS
C       -----------------
C
C       SET MAXIMUM NUMBER OF TIMES BEFORE POINTER SYSTEM IS SAVED
        MITRPS = IPASKY(27)

C       SET MAXIMUM NUMBER OF NGIVTI
        KHAFEZ = IPASKY(28)

C       MAXIMUM NUMBER OF ITERATIONS AFTER WHICH EPS1TI IS DECREASED
        MITEPS = IPASKY(32)

C       SEE IF YOU WANT TO USE A CURTAILED DOMAIN FOR INTEGRATION PURPOSE
C       AND/OR WANT TO CALCULATE DIFFUSION AFTER AFTER A SPECIFIED
C       NUMBER OF ITERATION (KBLOCK SHOULD BE MORE THAN 1)
        KBLOCK = IAND(IPASKY(40),1)
        NDIFFC = KBLOCK
        IF (IPASKY(40) .GT. KBLOCK) NDIFFC = IPASKY(40)

C       MINIMUM ERROR BELOW WHICH EPS1TI WILL BE INCREASED
        ERRMIN = APASKY(26)

C       MAXIMUM ERROR ABOVE WHICH EPS1TI WILL BE DECREASED
        ERRMTI = APASKY(27)

C       MINIMUM ALLOWABLE VALUE OF EPS1TI
        EPS1MN = APASKY(28)

C       MAXIMUM ALLOWABLE VALUE OF EPS1TI
        EPS1MX = APASKY(29)

C
C       ----------------------
C       NORMAL RUN STARTS HERE
C       ----------------------
C
C       SEE IF THE SCHEDULE PROGRAM IS NEEDED
10      CALL   E2SCHO

C       SET THE DIFFUSION COEFFICIENTS AT ALL NODES FOR FIRST TIME
        CALL E2DIFF
C
C       SEE IF A CURTAILED GRID IS TO BE USED
        IF (KBLOCK .NE. 0) CALL SHORTG(1)

20      NTERE  = NTERE  + 1
        NTERT  = NTERT  + 1
        NTERD  = NTERD  + 1
        NITRE2 = NITRE2 + 1

C       SEE IF THE POINTER SYSTEM IS TO BE SAVED; IF SO ONLY UNFORMATTED
C       OUTPUT WILL BE WRIITEN

        IF (NTERE .GE. MITRPS) THEN
          NTERE  = 0
          TDUM   = TIMNTI
```

```
          TIMNTI = TIME
          WRITE(JTERMO,*) ' WRITTING UNFORMATTED OUTPUT ON',JDUMY4
C         READ THE FULL DOMAIN AGAIN IF A CURTAILED DOMAIN WAS USED
          IF (KBLOCK .NE. 0) CALL SHORTG(2)
          CALL PSWRTU (JDUMY4)
          EPSLE2 = EPSOLD
          IDBGFR = IDBOLD
C         READ THE CURTAILED DOMAIN AGAIN
          IF (KBLOCK .NE. 0) CALL SHORTG(3)
          CLOSE (JDUMY4)
          TIMNTI = TDUM
          JDUMY4 = JDUMY4 + 1
          IF (JDUMY4 .EQ. 40) JDUMY4 = 34
        ENDIF
C
C
C       SEE IF SPATIAL ADAPTATION IS NEEDED
C       MITRA2 DENOTES THE NUMBER OF ITERATION (OR PASSES) AFTER
C       WHICH ADAPTATION IS DONE; ADAPTATION LOOP IS BY-PASSED
C       IF METHA2 = 0 (METHOD OF ADAPTATION)

        IF (METHA2 .NE. 0) THEN
           NTERA = NTERA + 1
           IF (NTERA  .EQ. 1     ) CALL  A2MTHO
           IF (NTERA  .GE. MITRA2) NTERA = 0
           IF (KPERFR .EQ. 1     ) CALL  TVINI1
           IF (KBLOCK .NE. 0) CALL SHORTG(1)
        ENDIF

C       SET ALL CHANGES TO ZERO FOR ALL THE NODES
        CALL E2ZERO

C       COMPUTE THE TIME STEPS FOR EACH CEWIC CELL
        CALL E2TIMO
        FCTRTI = 1.

C       COMPUTE THE NUMBER OF INTEGRATION PASSES
        KMAX   = 2**NMAXTI
        IPASSM = 2*KMAX - 1

        IF (KADPTI .EQ. 99) DTMNTI = 0.
C       COMPUTE THE CURRENT TIME OF THE RUN
C       IF UNSTEADY INLET BOUNDARY CONDITIONS ARE NOT USED THEN
C       UNCOMMENT THE FOLLOWING LINE AND COMMENT THE CONDITIONAL
C       STATEMENT INSIDE THE LOOP ITSELF FOR EFFICIENT CALCULATION
C       TIME  = TIME + KMAX*DTMNTI

C       DETERMINE THE NODES AT ALL TEMPORAL LEVELS
        CALL G2HANG

        DO 30 IPASS = 1, IPASSM
C          DETERMINE THE TEMPORAL LEVEL OF CELLS TO BE INTEGRATED
           ITGL = ITLEVL(IPASS,NMAXTI)

C          CALCULATE CHANGE AND DISTRIBUTE FOR ALL CELLS ON THIS LEVEL
           CALL E2SOLO (ITGL)
```

```
C          DETERMINE THE CURRENT TIME FOR UNSTEADY INLET CONDITIONS
           IF (ITGL .EQ. 0) THEN
             TIME = TIME + DTMNTI
C               SEE IF PERIODIC BOUNARY CONDITIONS ARE NEEDED
C               COMMENT THIS OUT IF SUCH CONDITIONS ARE NOT USED
                IF (KPERFR .EQ. 1) CALL E2VARB (TIME)
           ENDIF
C          APPLY BOUNDARY CONDITIONS
           CALL E2BCNO (ITGL)

C          COLLECT THE CONVERGENCE HISTORY IF NEED BE
           CALL E2CONO (TIME, ITGL, IPASS, IPASSM)

C          UPDATE ALL NODES AT THIS LEVEL
           CALL E2UPDO (ITGL)

C          LOOP BACK FOR NEXT TEMPORAL LEVEL CELLS

30      CONTINUE
C
C       ADD ARTIFICIAL SMOOTHING
C
C       SET THE DIFFUSION COEFFICIENTS AT ALL THE NODES
        IF (NTERD .GE. NDIFFC) THEN
           NTERD  = 0
           CALL E2DIFF
        ENDIF

C       SEE IF TEMPORAL ADJUSTMENTS ARE NEEDED
C       IF (MITRPS .NE. 0) THEN
C         IF (ERORE2 .LT. ERRMIN) THEN
C           NITEPS = NITEPS + 1
C           IF (NITEPS .GT. MITEPS) THEN
C              EPS1NW = 1.05*EPS1TI
C              EPS1TI = MIN (EPS1NW, EPS1MX)
C              NITEPS  = 0
C              IF (NGIVTI .LT. KHAFEZ) NGIVTI = NGIVTI + 1
C           ENDIF
C         ELSE
C           NITEPS = 0
C         ENDIF
CC
C         IF (ERORE2 .GT. ERRMTI) THEN
C           EPS1NW = 0.95*EPS1TI
C           EPS1TI = MAX (EPS1NW, EPS1TI)
C         ENDIF
C       ENDIF

        IF (NITRE2 .GE. MITRE2) GOTO 40
        IF (ERORE2 .LE. EPSLE2) GOTO 40
        IF (TIME   .GE. TIMXTI) GOTO 40
C       IF (NCYCFR .GE. MCYCFR+1) GOTO 40
        GO TO 20

40      CONTINUE

C       PRINT OUT PARAMETERS
```

```
C       IF (IDBGFL .EQ. 7 .OR. IDBGFL .GT. 1000)  THEN
C           WRITE(JDEBUG,1000)
C           WRITE(JDEBUG,1100)
C           WRITE(JDEBUG,1200)
C           WRITE(JDEBUG,1300) MITRPS, MITEPS, KTIMTI, MITRA2, NITRE2,
C     1                        KHAFEZ
C           WRITE(JDEBUG,1400) ERRMIN, ERRMTI, EPS1MN, EPS1MX, ERORE2,
C     1                        TIME  , EPS1TI
C       ENDIF
C1000   FORMAT(//10X,'----------------------' )
C1100   FORMAT(  10X,'DEBUG PRINT FROM TWODOU' )
C1200   FORMAT(  10X,'----------------------'/)
C1300   FORMAT(5X,'MITRPS = ',I5,10X,'MITEPS = ',I5/
C     1       5X,'KTIMTI = ',I5,10X,'MITRA2 = ',I5/
C     2       5X,'NITRE2 = ',I5,10X,'KHAFEZ = ',I5/)
C1400   FORMAT(5X,'ERRMIN = ',G14.5,10X,'ERRMTI = ',G14.5/
C     1       5X,'EPS1MN = ',G14.5,10X,'EPS1MX = ',G14.5/
C     2       5X,'ERORE2 = ',G14.5,10X,'TIME   = ',G14.5/
C     3       5X,'EPS1TI = ',G14.5,10X,                 /)

        TIMNTI = TIME
        IF (JREADS .EQ. 0) THEN
C          READ THE FULL DOMAIN AGAIN IF A CURTAILED DOMAIN WAS USED
           IF (KBLOCK .NE. 0) CALL SHORTG(2)
           CALL E2FINI
           STOP ' THE END'
        ENDIF

        GOTO 10

        END
```

WRINI2

```
        SUBROUTINE WRINI2

        INCLUDE 'PRECIS.INC '
        INCLUDE 'PARMV2.INC '
        INCLUDE 'A2COMN.INC '
        INCLUDE 'CHCOMN.INC '
        INCLUDE 'E2COMN.INC '
        INCLUDE 'FLCOMN.INC '
        INCLUDE 'G2COMN.INC '
        INCLUDE 'IOCOMN.INC '
        INCLUDE 'KYCOMN.INC '
        INCLUDE 'TICOMN.INC '

        CHARACTER CHARVA*32

C***********************************************************************

C       THIS SUBROUTINE WRITES THE INITIAL INFORMATION ABOUT THIS RUN
C       NCRSG2 IS NOT NEEDED HERE ******** REVISE LATTER
```

```
C*********************************************************************
      WRITE (JOUTAL,10) MTITLE
10    FORMAT('1',//,10X,A80/)

      IF (KSRTE2 .EQ. 0 .OR. KSRTE2 .EQ. 1000) THEN
        CHARVA = 'NEW RUN'
      ELSE
        CHARVA = 'RESTART'
      ENDIF

      WRITE (JOUTAL,20) NEQNFL, NEQSCH, NREACH, NSPECH, NINRCH,
     1           NXTDA2, MITRE2, KSRTE2, CHARVA, MALVG2
C     1           NXTDA2, MITRE2, KSRTE2, CHARVA, MALVG2, NCRSG2

20    FORMAT(5X,'TOTAL NUMBER OF EQUATIONS     =',I5,
     1       5X,'NUMBER OF SPECIES EQUATIONS   =',I5/
     2       5X,'NUMBER OF REACTIONS           =',I5,
     3       5X,'NUMBER OF SPECIES             =',I5/
     4       5X,'NUMBER OF INERT SPECIES       =',I5,
     5       5X,'NUMBER OF EXTENDED CELLS      =',I5/
     6       5X,'MAXIMUM ITERATIONS ALLOWED    =',I5,
     7       5X,'RUN STARTING PARAMETER        =',I5,5X,A8/
     8       5X,'MAXIMUM ALLOWED FINE LEVELS   =',I5       )
C    9       5X,'MAXIMUM ALLOWED COARSE LEVELS =',I5       )

      CHARVA                    = 'ERROR                    '
      IF (METHA2 .EQ. 0) CHARVA = 'NIL                      '
      IF (METHA2 .EQ. 1) CHARVA = 'NODE BASED VALUE         '
      IF (METHA2 .EQ. 2) CHARVA = 'CELL BASED VALUE         '
      IF (METHA2 .EQ. 3) CHARVA = 'NODE BASED FIRST GRADIENT'
      IF (METHA2 .EQ. 4) CHARVA = 'CELL BASED FIRST GRADIENT'
      IF (METHA2 .EQ. 6) CHARVA = 'CELL BASED MAX DIFFERNCE '

      WRITE (JOUTAL,30) METHA2, CHARVA
30    FORMAT(5X,'METHOD OF SPATIAL ADAPTATION  =',I5,5X,A32)

      IF (IMPLTI .EQ. 1) CHARVA(1:25) = 'EXPLICIT SOURCE TERMS    '
      IF (IMPLTI .EQ. 0) CHARVA(1:25) = 'IMPLICIT SOURCE TERMS    '
      WRITE (JOUTAL,35) IMPLTI, CHARVA
35    FORMAT(5X,'TYPE OF SOURCE TERM MODELLING =',I5,5X,A32)

      CHARVA          = 'ERROR                  '
      IF (K1ADA2 .LE. NEQNFL) THEN
         CHARVA(1:18) = 'DEPENDENT VARIABLE '
         WRITE(CHARVA(19:20),40) K1ADA2
40       FORMAT(I2)
         WRITE (JOUTAL,50) K1ADA2, CHARVA
50       FORMAT(5X,'SPATIAL ADAPTATION CRITERION  =',I5,5X,A32)
      ENDIF

      IF (K2ADA2 .LE. NEQNFL .AND. K2ADA2 .GT. 0) THEN
         CHARVA(1:18) = 'DEPENDENT VARIABLE  '
         WRITE(CHARVA(19:20),40) K2ADA2
         WRITE (JOUTAL,50) K2ADA2, CHARVA
      ENDIF
```

```
      CHARVA          = 'NIL                '

      IF (KADPTI .GT. 0 .AND. KADPTI .LE. NEQNFL) THEN
         CHARVA(1:18) = 'DEPENDENT VARIABLE   '
         WRITE(CHARVA(19:20),40) KADPTI
      ENDIF

      WRITE (JOUTAL,60) KADPTI, CHARVA
60    FORMAT(5X,'TEMPORAL RESOLUTION CRITERION =',I5,5X,A32)

      IF (KROGER .NE. 0) THEN
        IF (KROGER .EQ. 1) CHARVA = 'ROGER AND CHINITZ MODEL      '
        IF (KROGER .EQ. 2) CHARVA = 'LIGHT HILL DISSOCIATION MODEL'
        IF (KROGER .EQ. 3) CHARVA = 'FROZEN IDEAL GAS             '
        WRITE (JOUTAL,70) KROGER, CHARVA
70      FORMAT(5X,'TYPE OF CHEMISTRY MODEL       =',I5,5X,A32)
      ENDIF

      IF (KROGER .NE. 3) THEN
        WRITE (JOUTAL,80) (IS, IS = 1, NSPECH)
80      FORMAT(/5X, 'REACTION COEFFICIENTS FOR ALL SPECIES '/
     1          5X, 'REACTION', 2X, 'TYPE', 5X, 20I5        )

        DO 100 IR = 1, NREACH
          WRITE (JOUTAL,85)
          CHARVA = 'FORWARD '
          WRITE (JOUTAL,90) IR,CHARVA,(IALPCH(IS,IR), IS = 1, NSPECH)
          WRITE (JOUTAL,90) IR,CHARVA,(IALOCH(IS,IR), IS = 1, NSPECH)
          CHARVA = 'BACKWARD'
85        FORMAT(5X)
          WRITE (JOUTAL,90) IR,CHARVA,(IBETCH(IS,IR), IS = 1, NSPECH)
          WRITE (JOUTAL,90) IR,CHARVA,(IBTOCH(IS,IR), IS = 1, NSPECH)
90        FORMAT(5X, I5, 5X, A8, 1X, 20I5)
100     CONTINUE
      ENDIF

      IF (METHA2 .NE. 0) THEN
        WRITE (JOUTAL,110) ALPHA2, BETAA2, GAMMA2, DELTA2
110     FORMAT(/5X, 'SPATIAL ADAPTATION PARAMETERS'/
     1          5X, 'ALPHA2 = ',G10.5,5X,'BETAA2 = ',G10.5,
     2          5X, 'GAMMA2 = ',G10.5,5X,'DELTA2 = ',G10.5)
      ENDIF

      WRITE (JOUTAL,120) TREFCH, PRESCH
120   FORMAT(/5X,'REFERENCE CHEMISTRY TEMPERATURE AND PRESSURE'/
     1        5X,'TEMPERATURE = ',G10.5,10X,'PRESSURE = ',G10.5  )

      WRITE (JOUTAL,130) TREFFL, PRESFL, UGASFL, AMCHFL, DISTFL,
     1                   RHORFL, UREFFL, FMREFL, WDREFL

130   FORMAT(/5X,'REFERENCE FLUID QUANTITIES '/
     1        5X,'TEMPERATURE = ',G10.5,10X,'PRESSURE  = ',G10.5/
     2        5X,'GAS CONSTANT= ',G10.5,10X,'MACH NO   = ',G10.5/
     3        5X,'DISTANCE    = ',G10.5,10X,'DENSITY   = ',G10.5/
     4        5X,'VELOCITY    = ',G10.5,10X,'HT FORM   = ',G10.5/
     5        5X,'SOURCE TERMS= ',G10.5,10X                     )
```

```
      WRITE (JOUTAL,140) SMAXE2, SMINE2, EPSLE2, CFLNTI
140   FORMAT(/5X,'OTHER INFORMATION '/
     1         5X,'MAX VISCO   = ',G10.5,10X,'MIN VISCO  = ',G10.5/
     2         5X,'CONV CRIT   = ',G10.5,10X,'CFL NUMBER = ',G10.5)

      IF (KROGER .NE. 3) THEN
        WRITE (JOUTAL,150)
150     FORMAT(/5X,'ARHENIUS COEFICIENT FOR ALL THE REACTIONS'/
     1           5X,'REACTION',4X,'TYPE',9X,'PRE-EXPO',5X,
     2              'TEMP-EXPO',4X,'ENERGY')

        DO 170 IR = 1, NREACH
          WRITE (JOUTAL,85)
          CHARVA = 'FORWARD '
          WRITE (JOUTAL,160) IR, CHARVA, PREFCH(IR), EXPFCH(IR),
     1                       ENEFCH(IR)
160       FORMAT(5X, I5, 6X, A8, 2X, 3G14.5)
          CHARVA = 'BACKWARD'
          WRITE (JOUTAL,160) IR, CHARVA, PREBCH(IR), EXPBCH(IR),
     1                       ENEBCH(IR)
          CHARVA = 'EQUILIBR'
          WRITE (JOUTAL,160) IR, CHARVA, PREECH(IR), EXPECH(IR),
     1                       ENEECH(IR)
170     CONTINUE

        WRITE (JOUTAL,180)
180     FORMAT(/5X,'PROPERTIES OF ALL THE SPECIES'/
     1           5X,'SPECIES',5X,'CV',12X,'CP',11X,'HT FORM',6X,
     2              'MASS FRAC',6X,'MOL WT',8X,'ENTROPY',7X,'BS')

        DO 200 IS = 1, NSPECH
          WRITE (JOUTAL,190) IS, SPCVCH(IS), SPCPCH(IS), FMHTCH(IS),
     1                 YSPECH(IS), AMWTCH(IS), ENTRCH(IS), SPBSCH(IS)
190       FORMAT(5X, I5, 2X, 7G14.5)
200     CONTINUE
      ENDIF

      IF (KROGER .EQ. 2) THEN
        PHI    = APASKY(1)
        RHOD   = APASKY(2)
        ETA    = EXPFCH(1)
        THETD  = ENEFCH(1)
        TOETA  = TREFFL**ETA
        UNITCF = UREFFL/(TOETA*RHORFL*DISTFL)
        CFBMA  = UNITCF*PHI
        CF     = CFBMA*AMWTCH(1)
        WRITE (JOUTAL,210) THETD, RHOD, CF, PHI


210     FORMAT(/5X,'THETAD     =',G14.5,10X,'RHOD      =',G14.5/
     1           5X,'CF         =',G14.5,10X,'PHI       =',G14.5)
      ENDIF

      RETURN
      END
```

D.4 Utility Routines

This section contains information on the utility routines used in the codes GNBLOC and STAR.

D.4.1 Link information

The file ULT.COM contains link information for creating a library UL2LIB.

```
$ LIBRARY/CREATE   UL2LIB    ERRORM,   GAUSS2,   GAUSS3,   HEADER,-
                             IBASE2,   IMAGEI,   INSIDE,   INTERP,-
                             ITLEVL,   LINCRS,   TIMEIT,   TIMERR,-
                             WARNIN,   BBLSRT,   AST
```

D.4.2 Listing of routines

AST

```
C       This set of subroutines is VMS specific, and alows the code
C       to be interrupted by an AST (Control-C or Control-Y)
C       AST stands for asychronous system trap
C
C       The AST handler is initialized with INIT_ASTC and/or
C       INIT_ASTY which set the AST handlers for CTRL-C and CTRL-Y
C
C       After a trap is received, the appropriate flag in the
C       common block AST$$$ is set to true. It is up to the
C       main program to reset these flags and reinitialize the AST
C
        SUBROUTINE SET_ASTC
        COMMON /AST$$$/ ASTC$$,ASTY$$
        LOGICAL ASTC$$,ASTY$$
        INCLUDE '($iodef)'
        INTEGER SYS$ASSIGN,SYS$QIOW
        INTEGER*2 INPUT_CHANNEL
        INTEGER*4 CODE,set_c,set_y
        EXTERNAL DO$_AST
        STRUCTURE /IOSTAT_BLOCK/
        INTEGER*2 IOSTAT
        BYTE TRANSMIT,RECEIVE,CRFILL,LFFILL,PARITY,ZERO
        END STRUCTURE
        RECORD /IOSTAT_BLOCK/ IOSB
C
C
        ASTC$$ = .FALSE.
```

```
      STATUS = SYS$ASSIGN('SYS$INPUT',INPUT_CHAN,,)
      CODE = IO$_SETMODE .OR. IO$M_CTRLCAST
      STATUS = SYS$QIOW (,%VAL(INPUT_CHAN),%VAL(CODE),IOSB,
     &           ,,DO$_AST,ASTC$$,,,,)
      RETURN
      END
C
C
C
      SUBROUTINE SET_ASTY
      COMMON /AST$$$/ ASTC$$,ASTY$$
      LOGICAL ASTC$$,ASTY$$
      INCLUDE '($iodef)'
      INTEGER SYS$ASSIGN,SYS$QIOW
      INTEGER*2 INPUT_CHANNEL
      INTEGER*4 CODE
      EXTERNAL DO$_AST
      STRUCTURE /IOSTAT_BLOCK/
      INTEGER*2 IOSTAT
      BYTE TRANSMIT,RECEIVE,CRFILL,LFFILL,PARITY,ZERO
      END STRUCTURE
      RECORD /IOSTAT_BLOCK/ IOSB
C
C
      ASTY$$ = .FALSE.
      STATUS = SYS$ASSIGN('SYS$INPUT',INPUT_CHAN,,)
      CODE = IO$_SETMODE .OR. IO$M_CTRLYAST
      STATUS = SYS$QIOW (,%VAL(INPUT_CHAN),%VAL(CODE),IOSB,
     &           ,,DO$_AST,ASTY$$,,,,)
      RETURN
      END
C
C
C
      SUBROUTINE DO$_AST (ASTREC)
      LOGICAL ASTREC
      ASTREC = .TRUE.
      RETURN
      END
```

EQUCAL

```
      SUBROUTINE EQUCAL (T,THETAD,RHO,RHOD,ETRAT,RHORAT,ALPHAE)

C     ******************************************************************
C
C     THIS SUBROUTINE CALCULATES THE EQUILIBRIUM DEGREE OF DISSOCIATION
C     FOR A LIGHTHILL GAS.
C
C     ******************************************************************

      TRAT   = THETAD/T
      RHORAT = RHO/RHOD

      IF (TRAT .GT. 80.) THEN
```

```
        ETRAT = 0.
      ELSE
        ETRAT = EXP(-TRAT)
      ENDIF

      A      = ETRAT/RHORAT
      DISCRI = SQRT(A*A+4.*A)
      ALPHAE = (DISCRI-A)/2.

      RETURN
      END
```

ERRORM

```
      SUBROUTINE ERRORM (NERROR       ,          ITEXTR       ,
     1                   ITEXT1       ,          ZER1         ,
     2                   ITEXT2       ,          ZER2         ,
     3                   JPRINT       ,          ITEXTM       )
C
C  ********************************************************************
C  *                                                                  *
C  *   THIS ROUTINE PRINTS AN ERROR MESSAGE ON JTERMO AND JPRINT      *
C  *   NERROR  CONTAINS THE ERROR NUMBER                              *
C  *   ITEXTR  CONTAINS THE ROUTINE NAME                              *
C  *   ITEXT1  CONTAINS THE FIRST  VARIABLE NAME                      *
C  *   ITEXT2  CONTAINS THE SECOND VARIABLE NAME                      *
C  *   ITEXTM  CONTAINS THE ERROR MESSAGE                             *
C  *   ZER1  IS THE VALUE OF THE FIRST  VARIABLE                      *
C  *   ZER2  IS THE VALUE OF THE SECOND VARIABLE                      *
C  *   EXCEPT FOR THE COMPUTER IN QUESTION COMMENT ALL THE LINES      *
C  *   BETWEEN THE  ------  MARKERS FOR ALL THE COMPUTERS WHICH ARE   *
C  *   NOT BEING USED HERE.                                           *
C  *                                                                  *
C  ********************************************************************
C
C      -------------------------
C      VAX/VMS SYSTEM
       INCLUDE '[.INC] PRECIS.INC/LIST'
C      -------------------------
C
       CHARACTER*6   ITEXTR,   ITEXT1,   ITEXT2
       CHARACTER*(*) ITEXTM
C
       JTERMO = 6
C
       WRITE(JTERMO,10) NERROR,ITEXTR,ITEXTM,ITEXT1,ZER1,ITEXT2,ZER2
       WRITE(JPRINT,10) NERROR,ITEXTR,ITEXTM,ITEXT1,ZER1,ITEXT2,ZER2
C
C      GET A TRACEBACK; THERE IS NO TRACEBACK ON UNIX SYSTEMS
C
C      -------------------------
C      VAX/VMS SYSTEM
       CALL LIB$SIGNAL(%VAL(2))
C      -------------------------
C
```

```
C       -------------------------
C       JVNCC --- CYBER 205
C       TO BE DONE LATTER
C       -------------------------
        CALL EXIT
C
10      FORMAT(/' ERROR # ',I3,' DETECTED IN ROUTINE ',A6,/5X,A//
     1   1X,A6,' = ',G15.6,  5X,A6,' = ',G15.6//)
C
        RETURN
        END
```

GAUSS2

```
        SUBROUTINE GAUSS2 (A, R, X, IROW, IMAX)
        INCLUDE '[.INC] PRECIS.INC/LIST'
        DIMENSION A(IMAX,IMAX), R(IMAX), X(IMAX)

C***********************************************************************
C
C       THIS SUBROUTINE COMPUTES THE RESULT ( X ) OF
C                   A X = R
C       BY USING THE GAUSS ELIMINATION METHOD
C         ORDER OF A = IMAX x IMAX      COEFFICIENT MATRIX
C         ORDER OF X = IMAX x 1         VECTOR TO BE SOLVED
C         ORDER OF R = IMAX x 1         RHS VECTOR
C
C       THE MATRICES A AND R ARE CHANGED ON OUTPUT
C
C***********************************************************************
C
C       SAVE THE RHS VECTOR IN CASE OF ILL-CONDITIONED MATRIX
C
        DO 10 K = 1, IROW
          X(K) = R(K)
10      CONTINUE
C
        K      = 1
        IROWM1 = IROW - 1
C
C       NOTE THAT FOR ILL-CONDITIONED COEFFICIENT MATRIX THE RHS VECTOR
C       REMAINS THE SAME
C
20      IF (A(K,K) .EQ. 0.) RETURN
C
C       FIND THE NORMALIZING FACTOR FOR THE Kth ROW
C
        TEMP = 1./A(K,K)
C
C       NORMALIZE THE Kth ROW OF THE COEFFICIENT MATRIX
C
        DO 30 J  = K, IROW
          A(K,J) = A(K,J)*TEMP
```

```
30      CONTINUE
C
C       NORMALIZE THE Kth ROW OF THE RHS VECTOR
C
        R(K)    = R(K)*TEMP
        J       = K+1
40      TEMP    = A(J,K)
C
C       ZERO OUT THE Kth COLUMN OF ALL THE REMAINING ROWS
C
        DO 50 L  = K, IROW
          A(J,L) = A(J,L) - A(K,L)*TEMP
50      CONTINUE
C
C       APPLY THE SAME TRANSFORMATION ON THE RHS VECTOR
C
        R(J)    = R(J) - R(K)*TEMP
        IF(J .EQ. IROW) GOTO 60
        J       = J+1
        GOTO 40
60      IF (K .EQ. IROWM1) GOTO 70
        K       = K+1
        GOTO 20
70      IF (A(IROW,IROW) .EQ. 0.) RETURN
C
C       NOW DO REVERSE SUBSTITUTION
C
        X(IROW) = R(IROW)/A(IROW,IROW)
        I       = 1
80      SUM     = 0.
        J       = IROW - I + 1
90      SUM     = SUM + A(IROW-I,J)*X(J)
        IF (J .EQ. IROW) GOTO 100
        J       = J+1
        GOTO 90
100     L       = IROW - I
        X(L)    = R(L)-SUM
        IF ( I .EQ. IROWM1) GOTO 110
        I       = I+1
        GOTO 80
110     CONTINUE

        RETURN
        END
```

GAUSS3

```
        SUBROUTINE GAUSS3 (A, R, X, IROW, IMAX)
        INCLUDE '[.INC] PRECIS.INC/LIST'
        DIMENSION A(IMAX,IMAX), R(IMAX), X(IMAX)

C*******************************************************************
C
C       THIS SUBROUTINE COMPUTES THE RESULT ( X ) OF
C                    A X = R
```

```
C       BY USING THE GAUSS ELIMINATION METHOD
C         ORDER OF A = IMAX x IMAX     COEFFICIENT MATRIX
C         ORDER OF X = IMAX x 1        VECTOR TO BE SOLVED
C         ORDER OF R = IMAX x 1        RHS VECTOR
C
C       THE MATRICES A AND R ARE CHANGED ON OUTPUT
C
C*********************************************************************

        KI     = 5
C
C       NOTE THAT FOR ILL-CONDITIONED COEFFICIENT MATRIX THE RHS VECTOR
C       REMAINS THE SAME.  FURTHERMORE THE ROUTINE ASSUMES THAT THE FIRST
C       FOUR COMPONENTS ARE TRIVIAL
C
        DO 40  K = KI, IROW-1
C
           IF (A(K,K) .EQ. 0.) RETURN
C
C          FIND THE NORMALIZING FACTOR FOR THE Kth ROW
C
           TEMP = 1./A(K,K)
C
C          NORMALIZE THE Kth ROW OF THE COEFFICIENT MATRIX
C
           DO 10 J  = K, IROW
             A(K,J) = A(K,J)*TEMP
10         CONTINUE
C
C          NORMALIZE THE Kth ROW OF THE RHS VECTOR
C
           R(K)    = R(K)*TEMP
C
C          NOW TRANSFORM THE REMAINING ROWS
C
           DO 30 J = K+1, IROW
C
              TEMP    = A(J,K)
C
C             ZERO OUT THE Kth COLUMN OF ALL THE REMAINING ROWS
C
              DO 20 L  = K, IROW
                 A(J,L) = A(J,L) - A(K,L)*TEMP
20            CONTINUE
C
C             APPLY THE SAME TRANSFORMATION ON THE RHS VECTOR
C
              R(J)    = R(J) - R(K)*TEMP

30         CONTINUE

40      CONTINUE

        IF (A(IROW,IROW) .EQ. 0.) RETURN
C
C       NOW DO REVERSE SUBSTITUTION
C
```

```
      X(IROW) = R(IROW)/A(IROW,IROW)

      DO 60 I = 1, IROW-1
         SUM = 0.
         DO 50 J = IROW - I + 1, IROW
            SUM     = SUM + A(IROW-I,J)*X(J)
50       CONTINUE
         L       = IROW - I
         X(L)    = R(L)-SUM
60    CONTINUE

      RETURN
      END
```

GRAMSM

```
      subroutine gramsm (aLvect, dvect, ndimen, mdimen)

      implicit real*8 (a-h,o-z)
      dimension aLvect(mdimen,mdimen), dvect(mdimen)

c**********************************************************************

c     This subroutine computes the orthonormal set of vectors from a
c     given set of vectors stored in einvector matrix aLvect.  The
c     vectors are stored as rows in aLvect. (i.e., the jth vector
c     is aLvect(j,k) where k varies from 1 to ndimen).  The ortho-
c     normal set is also returned in aLvect.

c**********************************************************************

      do 70 is = 1, ndimen

c       initialize the summation dummy vector

        do 10 k = 1, ndimen
          dvect(k) = 0.
10      continue

        do 40 it = 1, is-1

c         determine the dot product of the sth and tth vectors

          sdot = 0.
          do 20 k = 1, ndimen
            sdot = sdot + aLvect(is,k)*aLvect(it,k)
20        continue

c         multiply this dot product by the tth vector

          do 30 k = 1, ndimen
            dvect(k) = dvect(k) + sdot*aLvect(it,k)
30        continue

40      continue
```

```
c         subtract off the non-orthogonal components (dvect) from
c         the sth vector

          snorm = 0.
          do 50 k = 1, ndimen
            aLvect(is,k) = aLvect(is,k) - dvect(k)
            snorm        = snorm + aLvect(is,k)*aLvect(is,k)
50        continue

c         normalize the sth vector
          snorm = sqrt(snorm)

          do 60 k = 1, ndimen
            aLvect(is,k) = aLvect(is,k)/snorm
60        continue

70      continue

        return
        end
```

HEADER

```
        SUBROUTINE HEADER (JUNIT, ITEXT, MTITLE)
C
C  ********************************************************************
C  *                                                                  *
C  *   THIS SUBROUTINE WRITES A HEADER ON UNIT JUNIT                  *
C  *   THE HEADER CONSISTS OF THE ACTUAL TITLE OF THE RUN AND THE     *
C  *   SPECIFIC EXTRA HEADING THAT MAY BE REQUESTED, IT ALSO PRINTS   *
C  *   THE CURRENT TIME AND DATE OF THE RUN.                          *
C  *   EXCEPT FOR THE COMPUTER IN QUESTION COMMENT ALL THE LINES      *
C  *   BETWEEN THE  ------  MARKERS FOR ALL THE COMPUTERS WHICH ARE   *
C  *   NOT BEING USED HERE.                                           *
C  *                                                                  *
C  ********************************************************************
C
C       -------------------------
C       VAX/VMS SYSTEM
        INCLUDE '[.INC] PRECIS.INC/LIST'
        CHARACTER     DATECH*9 , TIMECH*8
C       -------------------------
C
C       -------------------------
C       JVNCC --- CYBER 205
C       CHARACTER     DATECH*8 , TIMECH*8
C       -------------------------
C
C       -------------------------
C       ALLIANT SYSTEM --- ISAAC
C       CHARACTER     FDATE*24
C       -------------------------
C
        CHARACTER*(*) ITEXT    , MTITLE
```

```
C
C
C       GET THE DATE AND TIME VARIABLES
C
C       -------------------------
C       VAX/VMS SYSTEM
        CALL DATE (DATECH)
        CALL TIME (TIMECH)
        WRITE(JUNIT,10) MTITLE,DATECH,TIMECH,ITEXT
10      FORMAT('1',1X, A80, T100, 'ON: ', A9, ' AT: ', A8/ 1X, A //)
C       -------------------------
C
C       -------------------------
C       JVNCC --- CYBER 205
C       DATECH = DATE()
C       TIMECH = TIME()
C       WRITE(JUNIT,10) MTITLE,DATECH,TIMECH,ITEXT
C10     FORMAT('1',1X, A80, T100, 'ON: ', A8, ' AT: ', A8/ 1X, A //)
C       -------------------------
C
C       -------------------------
C       ALLIANT SYSTEM --- ISAAC
C       WRITE(JUNIT,10) MTITLE,FDATE(),ITEXT
C10     FORMAT('1',1X, A80, T100, 'ON: ', A24/1X, A //)
C       -------------------------
C
        RETURN
        END
```

IBASE2

```
        INTEGER FUNCTION IBASE2 (KK, MMAX)
        INCLUDE '[.INC] PRECIS.INC/LIST'

C***********************************************************************

C       THIS FUNCTION CALCULATES THE TEMPORAL LEVEL OF THE CELLS GIVEN
C       THE RATIO
C            K = DT(CELL)/DTMIN
C       TO AVOID TEMPORAL STIFFNESS THE MAXIMUM LEVEL IS LIMITED BY
C       MMAX, I.E., KMAX = 2**MMAX

C***********************************************************************

        ZBASE  = 2.
        ZKK    = KK
        ZZ     = LOG (ZKK) / LOG(ZBASE)
        N      = INT (ZZ)
        N      = MIN (N,MMAX)
        IBASE2 = N

        RETURN
        END
```

IMAGEI

```
      SUBROUTINE IMAGEI (JOUTPU, JINPUT, MTITLE)
C
      INCLUDE '[.INC] PRECIS.INC/LIST'
      CHARACTER     ICARD*80, ISTAR*1
      CHARACTER*(*) MTITLE
      DATA ISTAR    /'*'/
C
C*********************************************************************
C
C       THIS SUBROUTINE READS THE INPUT FILE  ON JINPUT AND PRINTS
C       AN IMAGE OF IT ON THE GIVEN UNIT JOUTPU, IT ALSO INITIALIZES
C       THE HEADER TITLE WHICH IS USED WHEN PRINTTING AND PLOTTING.
C
C*********************************************************************
C
C       READ THE TITLE
C
      REWIND JINPUT
C
      READ  (JINPUT, 10, END=80) MTITLE
10    FORMAT(A)
C
C       REWIND AND PRINT IMAGE OF THE FILE
C
      CALL HEADER (JOUTPU, 'image of input file', MTITLE)
C
      WRITE  (JOUTPU, 20)
20    FORMAT (11X,40H         1         2         3         4,
     1             40H         5         6         7         8/
     2          11X,40H1234567890123456789012345678901234567890,
     3             40H1234567890123456789012345678901234567890/)
C
      REWIND JINPUT
      NCARD  = 1
C
30    READ   (JINPUT, 40, END=60) ICARD
40    FORMAT (A)
C
      WRITE  (JOUTPU, 50) NCARD, ICARD
50    FORMAT (' card',I5, ':', A)
      NCARD  = NCARD + 1
      GO TO 30
C
C       WRITE OUT COLUMN HEADINGS AGAIN
C
60    WRITE (JOUTPU, 20)
C
C       REWIND AND REPOSITION FILE AFTER TITLE AND COMMENTS
C
      REWIND JINPUT
      READ  (JINPUT, 40) ICARD
70    READ  (JINPUT, 40) ICARD
      IF  (ICARD(1:1) .EQ. ISTAR) GO TO 70
      BACKSPACE JINPUT
```

```
C
      RETURN
C          -
C     NO INPUT FILE EXISTS
C
80    WRITE  (JTERMO, 90)
90    FORMAT (' INPUT FILE DOES NOT EXIST')
C
      RETURN
      END
```

INSIDE

```
      SUBROUTINE INSIDE (IIN, X, NSIDES, XX, YY)
      INCLUDE '[.INC] PRECIS.INC/LIST'
      DIMENSION X(2,*)

C*********************************************************************
C
C     THIS SUBROUTINE DETERMINES IF A POINT IS INSIDE A POLYGON OR NOT
C     THE BOX IS MADE UP OF THE SEGMENTS 1-2,2-3,...,NSIDES-1
C     XX,YY IS THE TEST POINT
C     X IS AN ARRAY CONTAINING THE COORDINATES OF THE VERTICES
C     NSIDES IS THE NUMBER OF POINTS IN THE POLYGON
C     IIN = 1 IF THE POINT IS IN THE BOX
C
C*********************************************************************

      IIN = 0
C
C     (XA,YA) IS THE UPPER RIGHT CORNER AND (XB,YB) IS THE LOWER LEFT
C     CORNER OF THE SMALLEST SQUARE CIRCUMSCRIBING THE POLYGON.
C
      XA = X(1,1)
      YA = X(2,1)

      DO 10 I = 2, NSIDES
          XA = MAX (XA,X(1,I))
          XB = MIN (XB,X(1,I))
          YA = MAX (YA,X(2,I))
          YB = MIN (YB,X(2,I))
10    CONTINUE
C
C     IF THE POINT IS NOT IN THE CIRCUMSCRIBING SQUARE THEN IT IS NOT
C     IN THE POLYGON
C
      IF (XX .LT. XB .OR. XX .GT. XA) RETURN
      IF (YY .LT. YB .OR. YY .GT. YA) RETURN
C
C     FIND A POINT GUARANTEED TO BE OUTSIDE THE BOX BY ADDING THE
C     VECTOR (XA,YA)-(XB,YB) TO (XA,YA)
C
      XA = XA + XA - XB
      YA = YA + YA - YB
C
```

```
C       NOW CHECK TO SEE HOW MAY SIDES THE LINE SEGMENT FROM (XX,YY) TO
C       (XA,YA) INTERSECTS. IF THERE ARE AN ODD NUMBER OF INTERSECTIONS,
C       (XX,YY) IS INSIDE THE BOX.
C
        INTRCT = LINCRS (X(1,1),X(2,1),X(1,NSIDES),X(2,NSIDES),
     1                   XX,YY,XA,YA,S,T)
        DO 20 I = 1,NSIDES - 1
            INTRCT = INTRCT + LINCRS (X(1,I),X(2,I),X(1,I+1),
     1                                X(2,I+1),XX,YY,XA,YA,S,T)
            IF (S .EQ. 0.) IIN = 1
20      CONTINUE
        IF (MOD(INTRCT,2) .EQ. 1) IIN = 1

        RETURN
        END
```

INTERP

```
        SUBROUTINE  INTERP (RECT, DPENVA, NEQNFL)

        INCLUDE '[.INC] PRECIS.INC/LIST'
        DIMENSION RECT(2,5), DPENVA(5,*)

C*****************************************************************************

C       THIS SUBROUTINE INTERPOLATES THE DEPENDENT VARIABLES AT A POINT
C       INTERIOR TO A GIVEN QUADRILATERAL.  THE CORNERS OF THE QUADRI-
C       LATERAL ARE STORED IN (X1,Y1) THROUGH (X4,Y4).  THE TEST POINT
C       IS (X5,Y5).  NOTE THAT X = RECT(1,*) ETC.

C*****************************************************************************
C
C       DETERMINE THE COEFFICIENTS OF THE RHS MATRIX
C
        A11 = RECT(1,2) - RECT(1,1)
        A21 = RECT(1,3) - RECT(1,1)
        A31 = RECT(1,4) - RECT(1,1)
        A12 = RECT(2,2) - RECT(2,1)
        A22 = RECT(2,3) - RECT(2,1)
        A32 = RECT(2,4) - RECT(2,1)
        A13 = A11*A12
        A23 = A21*A22
        A33 = A31*A32
C
C       DETERMINE THE DISTANCES FROM THE TEST POINT
C
        DX   = RECT(1,5) - RECT(1,1)
        DY   = RECT(2,5) - RECT(2,1)
        DXDY = DX*DY
C
C       NOW DETERMINE ALL THE 2 x 2 DETERMINANTS
C
        D11 = A22*A33 - A23*A32
        D21 = A12*A33 - A13*A32
        D31 = A12*A23 - A13*A22
```

```
      D12 = A21*A33 - A23*A31
      D22 = A11*A33 - A13*A31
      D32 = A11*A23 - A13*A21
      D13 = A21*A32 - A22*A31
      D23 = A11*A32 - A12*A31
      D33 = A11*A22 - A12*A21

      DET = A11*D11 - A21*D21 + A31*D31
      DET = 1./DET
C
C     DETERMINE THE COEFFICIENTS A, B, C OF THE EQUATION
C
      DO 10 I = 1, NEQNFL
C
C        DETERMINE THE RHS VECTOR
C
         R1 = DPENVA(2,I) - DPENVA(1,I)
         R2 = DPENVA(3,I) - DPENVA(1,I)
         R3 = DPENVA(4,I) - DPENVA(1,I)
C
         AA = (R1*D11 - R2*D21 + R3*D31)*DET
         BB =-(R1*D12 - R2*D22 + R3*D32)*DET
         CC = (R1*D13 - R2*D23 + R3*D33)*DET
C
C        NOW COMPUTE THE DEPENDENT VARIABLES
C
         DPENVA(5,I) = DPENVA(1,I) + AA*DX + BB*DY + CC*DXDY

10    CONTINUE

      RETURN
      END
```

ITLEVL

```
      INTEGER FUNCTION ITLEVL (IPASS,  MMAX)
      INCLUDE '[.INC] PRECIS.INC/LIST'

C*********************************************************************

C     THIS FUNCTION CALCULATES THE TEMPORAL LEVEL OF THE CELLS TO
C     TO INTEGRATED AT PASS IPASS

C*********************************************************************

      IBASE  = 2

      DO 10 N = 0, MMAX
        IP2NP1 = IBASE**(N + 1)
        IP2N   = IP2NP1/2
        INTEG  = (IPASS - IP2N)/IP2NP1
        ITEST  = INTEG*IP2NP1 + IP2N

        IF (ITEST .EQ. IPASS) THEN
          ITLEVL = N
```

```
          RETURN
        ENDIF
          -
10      CONTINUE

C       NO LEVEL IS FOUND

        ZER1 = IPASS
        ZER2 = MMAX

        CALL ERRORM (23,'ITLEVL','IPASS ',ZER1,'NMAXTI',ZER2,JPRINT,
     1       'ERROR IN TEMPORAL LEVEL CALCULATION')

        RETURN
        END
```

LINCRS

```
        INTEGER FUNCTION LINCRS (X1,Y1,X2,Y2,X3,Y3,X4,Y4,S,T)
        INCLUDE '[.INC] PRECIS.INC/LIST'
        PARAMETER (EPSILON = 1.E-20)

C***********************************************************************
C
C       FUNCTION TO DETERMINE IF THE LINE SEGMENTS
C         (X1,Y1)--(X2,Y2) AND (X3,Y3)--(X4,Y4) CROSS
C
C       THIS IS DONE BY COMPUTING THE PARAMETERIZED INTERSECTION
C       AND SEEING IF THE PARAMETERS FOR BOTH LINES ARE IN THE
C       INTERVAL [0,1]. USE CRAMER'S RULE TO SOLVE THE  EQUATIONS
C             (X2-X1) T + (X3-X4) S = (X3 - X1)
C             (Y2-Y1) T + (Y3-Y4) S = (Y3 - Y1)
C
C       IF THERE IS NO SOLUTION, THE LINES ARE PARALLEL SO THEY
C       DO NOT CROSS ANYWAY.
C
C       S AND T ARE THE PARAMETERS OF THE CROSSING ON EACH LINE SEGMENT
C       S GOES FROM 0 TO 1 AS WE GO FROM 3 - 4
C       T GOES FROM 0 TO 1 AS WE GO FROM 1 - 2
C
C       LINCRS = 1 IF THE LINES CROSS, 0 OTHERWISE
C
C***********************************************************************
C
        LINCRS = 0
        S      = 0.
        T      = 0.
        XX1    = X2 - X1
        XX2    = X3 - X4
        XXX    = X3 - X1
        YY1    = Y2 - Y1
        YY2    = Y3 - Y4
        YYY    = Y3 - Y1
        DET    = XX1*YY2 - XX2*YY1
        IF (ABS(DET) .LT. EPSILON) RETURN
```

```
      T      = (XXX*YY2 - XX2*YYY) / DET
      S   -  = (XX1*YYY - XXX*YY1) / DET
C          -
C     CHECK TO SEE IF THERE IS AN INTERSECTION WITHIN THE PARAMETER
C     RANGES [0,1]
C
      IF (S .GE. 0. .AND. S .LE. 1. .AND.
     1    T .GE. 0. .AND. T .LE. 1.) LINCRS = 1
C
      RETURN
      END
```

TIMEIT

```
      SUBROUTINE TIMEIT(TIME)
C
      INTEGER*4 ITMLST(4),IRETLN,ICPUTM,IDELTM(2)
      INTEGER*2 IWORD(2),ITIMBF(7)
      EQUIVALENCE (IWORD(1),ITMLST(1))
C
C**********************************************************************
C
C     THIS SUBROUTINE LOOKS UP THE CURRENT CPU TIME IN SECONDS
C     THIS WAS ORIGINALLY WRITTEN BY J. DANNENHOFFER
C
C**********************************************************************
C
C
C     SET UP ITEM LISTS FOR CALL TO JOB/PROCESS INFO ROUTINES
C
      IWORD(1)=4
      IWORD(2)='0407'X
C
C     COMPUTE THE ADDRESS OF THE STORAGE ELEMENT (ICPUTM) AS AN INTEGER
C
      ITMLST(2)=%LOC(ICPUTM)
      ITMLST(3)=%LOC(IRETLN)
      ITMLST(4)=0
C
C     CALL SYSTEM SERVICE ROUTINES
C
      CALL SYS$GETJPI(,,,ITMLST,,,)
      CALL LIB$EMUL(ICPUTM,-100000,0,IDELTM)
      CALL SYS$NUMTIM(ITIMBF,IDELTM)
C
C     COMPUTE TIME
C
      TIME=86400.00*ITIMBF(3)
     1     + 3600.00*ITIMBF(4)
     1     +   60.00*ITIMBF(5)
     1     +    1.00*ITIMBF(6)
     1     +    0.01*ITIMBF(7)
C
C     ITIMBF(3)  IS THE CPU TIME IN DAYS
C     ITIMBF(4)  IS THE CPU TIME IN HOURS
```

```
C       ITIMBF(5)  IS THE CPU TIME IN MINUTES
C       ITIMBF(6)  IS THE CPU TIME IN SECONDS
C       ITIMBF(7)  IS THE CPU TIME IN DECI-SECONDS
C
        RETURN
        END
```

TIMERR

```
        SUBROUTINE TIMERR (JUNIT, TCUM, ITEXT)
C
C  ********************************************************************
C  *                                                                  *
C  *   THIS SUBROUTINE PRINTS A MESSAGE GIVEN BY ITEXT ON UNIT JUNIT.  *
C  *   IT ALSO PRINTS INCREMENTAL AND TOTAL CPU TIMES.  EXCEPT FOR THE *
C  *   COMPUTER IN QUESTION COMMENT ALL THE LINES BETWEEN THE ------  *
C  *   MARKERS FOR ALL THE COMPUTERS WHICH ARE NOT BEING USED HERE.    *
C  *                                                                  *
C  ********************************************************************
C
C       -------------------------
C       ISAAC --- ALLIANT COMPUTER
C       REAL TT1(2)
C       -------------------------
C
        CHARACTER ITEXT*(*)
        SAVE RESTAR, TSAVE, TSTART
        DATA RESTAR/0./
C
C       GET THE CPU TIME
C
C       -------------------------
C       ISAAC --- ALLIANT COMPUTER
C       T1   = ETIME(TT1)
C       TIME = TT1(1)
C       -------------------------
C
C       -------------------------
C       VAX/VMS SYSTEM
        CALL TIMEIT (TIME)
C       -------------------------
C
C       -------------------------
C       JVNCC --- CYBER 205
C       TIME = SECOND ()
C       -------------------------
C
C       CHECK IF INITIALIZATION
C
        IF (LEN(ITEXT) .LE. 1) THEN
            TSTART = TIME
            TSAVE  = 0.
            RETURN
        ENDIF
C
```

```
C       NORMAL PROCESSING
C       TCUM IS CUMMULATIVE TIME
C       TINC IS INCREMENTAL TIME
C
C       FOR RESTART RUNS TCUM IS READ FROM PSREAD FILE
C
        IF (ITEXT .EQ. 'RESTART') THEN
            RESTAR = TCUM
            RETURN
        ENDIF
C
        TIME   = TIME + RESTAR
        TCUM   = TIME - TSTART
        TINC   = TCUM - TSAVE
        TSAVE  = TCUM
C
C       ENCODE/DECODE
C
        IF(ITEXT .EQ. '.RETURN.') THEN
           WRITE(ITEXT,1000) TCUM
        ELSE
           WRITE(JUNIT,1100) TINC, TCUM, ITEXT
        ENDIF
C
1000    FORMAT(F8.2)
1100    FORMAT(' TIMER -- CPU INCREMENT =', G14.5, ' SEC', 5X,
     1         ' TOTAL CPU = ', G14.5, ' SEC', 5X, A         )

        RETURN
        END
```

WARNIN

```
        SUBROUTINE WARNIN (NERROR      ,          ITEXTR         ,
     1                     ITEXT1      ,          ZER1           ,
     2                     ITEXT2      ,          ZER2           ,
     3                     JPRINT      ,          ITEXTM         )
        DATA KOUNT/0/
        SAVE KOUNT
C
C  ********************************************************************
C  *                                                                  *
C  *   THIS ROUTINE PRINTS A WARNING MESSAGE ON JTERMO AND JPRINT     *
C  *   NERROR  CONTAINS THE ERROR NUMBER                              *
C  *   ITEXTR  CONTAINS THE ROUTINE NAME                              *
C  *   ITEXT1  CONTAINS THE FIRST  VARIABLE NAME                      *
C  *   ITEXT2  CONTAINS THE SECOND VARIABLE NAME                      *
C  *   ITEXTM  CONTAINS THE ERROR MESSAGE                             *
C  *   ZER1  IS THE VALUE OF THE FIRST  VARIABLE                      *
C  *   ZER2  IS THE VALUE OF THE SECOND VARIABLE                      *
C  *   EXCEPT FOR THE COMPUTER IN QUESTION COMMENT ALL THE LINES      *
C  *   BETWEEN THE  ------  MARKERS FOR ALL THE COMPUTERS WHICH ARE   *
```

```
C  *   NOT BEING USED HERE.                                                *
C  *       -                                                              *
C  ************************************************************************
C
C       -------------------------
C       VAX/VMS SYSTEM
        INCLUDE '[.INC] PRECIS.INC/LIST'
C       -------------------------
C
        CHARACTER*6   ITEXTR,   ITEXT1,   ITEXT2
        CHARACTER*(*) ITEXTM
C
        JTERMO = 6
        JDEBUG = 24
        KOUNT  = KOUNT + 1
        IF (KOUNT .GT. 10) RETURN
C
        WRITE(JDEBUG,10) NERROR,ITEXTR,ITEXTM,ITEXT1,ZER1,ITEXT2,ZER2
        WRITE(JTERMO,10) NERROR,ITEXTR,ITEXTM,ITEXT1,ZER1,ITEXT2,ZER2
        WRITE(JPRINT,10) NERROR,ITEXTR,ITEXTM,ITEXT1,ZER1,ITEXT2,ZER2
C
C       GET A TRACEBACK; THERE IS NO TRACEBACK ON UNIX SYSTEMS
C
C       -------------------------
C       VAX/VMS SYSTEM
        CALL LIB$SIGNAL(%VAL(2))
C       -------------------------
C
C       -------------------------
C       JVNCC --- CYBER 205
C       TO BE DONE LATTER
C       -------------------------
C
10      FORMAT(/' WARNING  # ',I3,' DETECTED IN ROUTINE ',A6,5X,A//
     1   1X,A6,' = ',G15.6,  5X,A6,' = ',G15.6//)

        RETURN
        END
```

D.5 GRAFIC Interface Routines

This section contains information on the GRAFIC interface routines called elsewhere in the previous sections. The actual routines in GRAFIC are listed here.

D.5.1 Link information

The file PLT.COM contains link information for these files.

```
$ LINKP2ALLP :==  LINK P2ALLP,   PLXSET,    ZRDUMY,    ZRPLTC,-
                                 ZRPLTG,    ZRVECT,    ZRPLTL,-
                                 [PERVAIZ.STAR.OBJ]PSRED2,-
                                 [PERVAIZ.STAR.OBJ]PSREDU,-
                                 [PERVAIZ.ULT.OBJ]UL2LIB/LIB,-
                                 [PERVAIZ.GRAFIC1]NEW_GRAFIC/LIB
$ LINKP2GRID :== LINK P2GRID,    ZRPLTG,-
                                 [PERVAIZ.STAR.OBJ]PSREDU,-
                                 [PERVAIZ.ULT.OBJ]UL2LIB/LIB,-
                                 [PERVAIZ.GRAFIC1]NEW_GRAFIC/LIB
$ LINKP2ITER :== LINK P2ITER,    PLXSET,-
                                 [PERVAIZ.GRAFIC1]NEW_GRAFIC/LIB
$ EXIT
```

D.5.2 Listing of routines

BINPNT

```
      PROGRAM BINPNT

      INCLUDE '[PERVAIZ.TWODO.INC]PRECIS.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] A2COMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] CHCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] E2COMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] FLCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] IOCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PRCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] TICOMN.INC /LIST'
      DIMENSION ZX(MNODG2), ZY(MNODG2), ZDPEN(MEQNFL,MNODG2),
     1          ZP(MNODG2), ZT(MNODG2), ZS(MNODG2)
      CHARACTER YESNO*1

C*********************************************************************
```

```
C       THIS SUBROUTINE READS ALL THE INFORMATION ABOUT THE POINTER
C       SYSTEM AND ALL THE OTHER ARRAYS FROM UNIT 'JPNTRE', FROM A
C       PREVIOUSLY RUN CASE.  THIS ROUTINE IS USEFUL WHEN YOU WANT TO
C       TAKE A SECOND LOOK AT SUCH A CASE FOR PLOTTING PURPOSES OR WHEN
C       YOU WANT TO GENERATE THE INITIAL CONDITIONS AGAIN.

C***********************************************************************

        JTERMI = 5
        JTERMO = 6
        JPLOTI = 19
        JPNTRE = 28
        JREADG = 29
        JREADD = 30
        TIMNTI = 1.

        WRITE (JTERMO,1000)
        READ  (JTERMI,*) ITYPE

        IF (ITYPE .EQ. 2) THEN
          CALL PSRED2
        ELSE
          CALL PSREDU
        ENDIF

        DO 20 IN = 1, NNODG2
          ZX(IN) = GEOMG2(1,IN)
          ZY(IN) = GEOMG2(2,IN)
          ZP(IN) = PRESG2(IN)
          ZT(IN) = TEMPG2(IN)
          ZS(IN) = SIGGE2(IN)
          DO 20 IQ = 1, NEQNFL
             ZDPEN(IQ,IN) = DPENG2(IQ,IN)
10        CONTINUE
20      CONTINUE

C
        IF (ITYPE .EQ. 1) THEN
          OPEN (UNIT=JREADG, FILE='INPUTGG.DAT', STATUS='NEW')
          OPEN (UNIT=JREADD, FILE='INPUTDD.DAT', STATUS='NEW')
          NXRECT = 0
          NYRECT = 0
          DO 30 IBN = 1, NBNDG2
            IF (IBNDG2(4,IBN) .EQ. 4) THEN
               NXRECT = IBN
               GOTO 40
            ENDIF
30        CONTINUE
40        WRITE (JTERMO,1100)
          READ  (JTERMI,1200) YESNO
          IF (YESNO .EQ. 'N' .OR. YESNO .EQ. 'n') THEN
            NYRECT = NNODG2/NXRECT
            GOTO 70
          ENDIF
          DO 60 IBN = NXRECT, NBNDG2
            NYRECT = NYRECT + 1
            IF (IBNDG2(4,IBN) .EQ. 6) GOTO 60
```

```
50        CONTINUE
60        NNODG2 = NXRECT*NYRECT
          NBNDG2 = 2*(NXRECT + NYRECT - 2)
C
C         WRITE ALL THE INFORMATION ON INPUTGG.DAT SO THAT IT CAN BE
C         READ BY G2INIT LATTER ON; NOTE THAT THE BASE NODES REMAIN
C         THE SAME WHETHER ADAPTATION WAS DONE OR NOT

70        WRITE (JREADG,1300) NXRECT, NYRECT, NBNDG2, NNODG2
          WRITE (JREADG,1300) (IBNDG2(5,IB), IB=1,NBNDG2)
          WRITE (JREADG,1400) (GEOMG2(1,IN),GEOMG2(2,IN), IN=1,NNODG2)
          DO 80 IN = 1, NNODG2
             WRITE (JREADD,1500) (DPENG2(K,IN), K = 1, NEQNFL)
80        CONTINUE
        ENDIF
C
        IF (ITYPE .EQ. 2 .OR. ITYPE .EQ. 3) THEN
          OPEN (UNIT=JPLOTI, FILE='JPLOTI.DAT', STATUS='NEW',
     1          FORM='UNFORMATTED')
          WRITE (JPLOTI) NNODG2, NEQNFL, NEQBAS, NEQSCH, NCELA2,
     1                   NCELG2, NBNDG2, NLVLG2
          WRITE (JPLOTI) GAMAFL, YNRTCH, TIMNTI, TREFFL, RHORFL
          DO 90 IN = 1, NNODG2
            WRITE(JPLOTI) ZX(IN),ZY(IN),ZP(IN), ZT(IN), ZS(IN),
     1        (ZDPEN(J,IN), J = 1, NEQNFL)
90        CONTINUE
          DO 100 IC = 1, NCELG2
            WRITE(JPLOTI) (ICELG2(IP,IC),IP=1,10)
100       CONTINUE
          WRITE(JPLOTI) (KAUXG2(IC),IC=1,NCELG2)
          WRITE(JPLOTI) (ICELA2(IC),IC=1,NCELA2)
          DO 110 IC = 1, NBNDG2
            WRITE(JPLOTI) (IBNDG2(IP,IC),IP=1,5)
110       CONTINUE
          WRITE(JPLOTI) (MTITLE(I:I),I=1,79)
          DO 120 IC = 1, NNODG2
            WRITE(JPLOTI) (NEIBG2(IP,IC),IP=1,4)
120       CONTINUE
        ENDIF
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------

1000    FORMAT( 5X, 'INPUT ONE OF THE FOLLOWING OPTION :'/
     1   10X,'1. REDO  INITIAL CONDITIONS'/
     2   10X,'2. GENERATE PLOTTING DATA FILE FROM FORMATTED'/
     3   10X,'3. GENERATE PLOTTING DATA FILE FROM UNFORMATTED'/)
1100    FORMAT( 5X,'HAS THE GRID BEEN PREVIOUSLY ADAPTED ? [Y/N]')
1200    FORMAT(A1)
1300    FORMAT(12I5)
1400    FORMAT(4G16.7)
1500    FORMAT(8G15.7)

        STOP
        END
```

P2ALLP

```
      PROGRAM P2ALLP

      PARAMETER (MCURVE = 13)
      INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] A2COMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] CHCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] E2COMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] FLCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] HEXCOD.INC     '
      INCLUDE '[PERVAIZ.TWODO.INC] IOCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] PRCOMN.INC /LIST'
      INCLUDE '[PERVAIZ.TWODO.INC] TICOMN.INC /LIST'

      DIMENSION ZX(MNODG2), ZY(MNODG2), ZF(MNODG2), E1TAX$(MCURVE)
      DIMENSION KKOPT(1), KN$(1), IPOINT(MCELG2), IMARKN(MCELG2)
      REAL*4    GRDUMY(30), ALIMITS(6), FRACTN(MEQNFL),
     1                XYPLOT(MCURVE,MNODG2),  CONT(60),
     2                XMIN,XMAX,YMIN,YMAX,UMIN,UMAX,VMIN,VMAX
C
       common /ast$$$/ astc$$, asty$$
       logical astc$$, asty$$

      CHARACTER PLTITL*96, YESNO*1, IDATE*9, ITIME*8, ISTRING*80,
     1                E1TAX$*8 , DEVNAM*5, PLOT_TYPE*15
      EXTERNAL ZRPLTL, ZRPLTC, ZRDUMY, ZRPLTG, ZRVECT

      DATA E1TAX$/'DENSITY ' ,  'U VELO  ' ,  'V VELO  ' ,
     2             'T ENERGY' ,  'PRESSURE' ,  'TEMPERAT' ,
     3             'ENTHALPY' ,  'MACH NO.' ,  'MASFRAC1' ,
     4                  'MASFRAC2' ,  'MASFRAC3' ,  'MASFRAC4' ,
     5                  'MASFRAC5'                              /

C*********************************************************************

C     THIS PROGRAM READS ALL THE INFORMATION ABOUT THE POINTER SYSTEM
C     AND ALL THE OTHER ARRAYS FROM UNIT 'JPNTRE', FROM A RUN CASE FROM
C     FILE JPNTRE.DAT.  THIS PROGRAM THEN MAKES THE VARIOUS KINDS OF
C     PLOTS FOR THE TWO-DIMENSIONAL CASE.  THE PLOTS CAN BE
C          1. COLOR  CONTOURS
C          2. LINE   CONTOURS
C          3. LINE   PLOTS
C          4. VECTOR PLOTS
C          5. GRID   PLOTS
C          6. VALUE INFORMATION
C     THE MAXIMUM NUMBER FOR SPECIES FRACTIONS IS 5.
C     TIME     = SQRT(GAMAFL)*TIME
C
C     THE LINE PLOTTER GENERATES A STRING OF DATA BY MARCHING
C     THROUGH THE FIELD EITHER TO THE NORTH OR TO THE EAST,
C     STARTING AT IABS(ISTART)
C     IF (ISTART .GT. 0) MARCH TO THE EAST
C     IF (ISTART .LT. 0) MARCH TO THE NORTH
```

```
C
C      THIS VECTOR PLOTTER GENERATES DATA ON ONE OF THE
C      FOLLOWING TYPES OF NODES, AS INDICATED BY NODTYP
C           -2.  CORNER NODES OF LEVEL -2 CELLS
C           -1.  CORNER NODES OF LEVEL -1 CELLS
C            0.  CORNER NODES OF BASE LEVEL CELLS
C            1.  CORNER NODES OF LEVEL 1 CELLS AND ALL BASE NODES
C           99.  ALL THE NODES
C
C*********************************************************************
C
       KTERMI = 5
       KTERMO = 6
       MTITLE = ' '
       PLTITL = ' '
       JPRINT = 7
       ITYPE  = 0
       JPNTRE = 28
       JPLOTD = 57

       ITYPE  = 1
       NODTYP = 1
       INCMIN = 44
       INCMAX = 225
       IBKGRN = 1

C
C      THE DEFAULT PARAMETERS OF THIS RUN ARE READ FROM UNIT 57
C
c      OPEN (UNIT=JPLOTD, FILE='[PERVAIZ.PLT.OBJ]DEFAULT.DAT',
       OPEN (UNIT=JPLOTD,
     1            FILE='ernst::sys$user:[PERVAIZ.PLT.OBJ]DEFAULT.DAT',
     1            STATUS='OLD',FORM='FORMATTED', READONLY, ERR=11)

C      WRITE (KTERMO,10)
C10    FORMAT( 5X, 'INPUT ONE OF THE FOLLOWING OPTION :'/
C    2        10X,'1. GENERATE PLOTTING DATA FILE FROM UNFORMATTED'/
C    3        10X,'2. GENERATE PLOTTING DATA FILE FROM FORMATTED'/)
C
C      READ VALUES FOR FORMATTED/UNFORMATTED FILE TYPE; TYPE OF DATE
C      CHARACTERS; MAXIMUM AND MINIMUM INDICES FOR COLOR CONTOURS; AND
C      THE BACKGROUND COLOR OF THE SCREEN
C
       READ (JPLOTD,*) ITYPE
       READ (JPLOTD,*) NODTYP
       READ (JPLOTD,*) INCMIN
       READ (JPLOTD,*) INCMAX
       READ (JPLOTD,*) IBKGRN
C
C      READ THE POINTER SYSTEM INFORMATION
C
11     IF (ITYPE .EQ. 2) THEN
          WRITE(KTERMO,*) ' READING FROM FORMATTED PLOTTING FILE'
          OPEN (UNIT=JPNTRE, FILE='JPNTRE.DAT', STATUS='OLD',
     1          FORM='FORMATTED', READONLY)
         CALL PSRED2
       ELSE
```

```
          WRITE(KTERMO,*) ' READING FROM UNFORMATTED PLOTTING FILE'
          OPEN (UNIT=JPNTRE, FILE='JPNTRE.DAT', STATUS='OLD',
     1       -   FORM='UNFORMATTED', READONLY)
          CALL PSREDU
        ENDIF
C
C       SAVE THE CPU TIME FOR THIS RUN
        ZCUM = WORKA2(3)

        DO 20 IN = 1, NNODG2
          ZX(IN) = GEOMG2(1,IN)
          ZY(IN) = GEOMG2(2,IN)
20      CONTINUE
C
        YUPPER = 1. - YNRTCH
        NEQSP1 = NEQSCH + 1

C       SETUP THE XYPLOT ARRAY

        DO 40 I = 1, NNODG2

           SOUND        = GAMAFL*PRESG2(I)/DPENG2(1,I)
           SOUND        = ABS(SOUND)
           DPENG2(2,I)  = DPENG2(2,I)/DPENG2(1,I)
           DPENG2(3,I)  = DPENG2(3,I)/DPENG2(1,I)
           VELO2        = DPENG2(2,I)**2 + DPENG2(3,I)**2
           SUMY         = 0.

           DO 30 IS = 1, NEQSCH
             JS         = IS + NEQBAS
             FRACTN(IS) = DPENG2(JS,I)/DPENG2(1,I)
             SUMY       = SUMY + FRACTN(IS)
30         CONTINUE

           IF (NEQSP1 .LE. 5) THEN
             FRACTN(NEQSP1)   = YUPPER - SUMY
             FRACTN(NEQSP1)   = MAX (0., FRACTN(NEQSP1))
             FRACTN(NEQSP1+1) = YNRTCH
           ENDIF

C          NOW SET THESE VALUES IN THE XYPLOT ARRAY

           XYPLOT( 1,I) = DPENG2(1,I)
           XYPLOT( 2,I) = DPENG2(2,I)
           XYPLOT( 3,I) = DPENG2(3,I)
           XYPLOT( 4,I) = DPENG2(4,I)
           XYPLOT( 5,I) = PRESG2(I)
           XYPLOT( 6,I) = TEMPG2(I)
           XYPLOT( 7,I) = (DPENG2(4,I)+PRESG2(I))/DPENG2(1,I)
           XYPLOT( 8,I) = SQRT(VELO2/SOUND)
           XYPLOT( 9,I) = FRACTN(1)
           XYPLOT(10,I) = FRACTN(2)
           XYPLOT(11,I) = FRACTN(3)
           XYPLOT(12,I) = FRACTN(4)
           XYPLOT(13,I) = FRACTN(5)
           PRESG2(I)    = DPENG2(2,I)
           TEMPG2(I)    = DPENG2(3,I)
```

```
40     CONTINUE
C
C      INITIALIZE THE GRAPHICS ROUTINES
C
       WRITE(KTERMO,50) NNODG2, NCELA2, NCELG2, NBNDG2, NLVLG2
50     FORMAT(5X,'NNODG2 =',I7,5X,'NCELA2 =',I7,5X,'NCELG2 =',I7/
     1        5X,'NBNDG2 =',I7,5X,'NLVLG2 =',I7)

       WRITE(KTERMO,60) TIMNTI, MTITLE
60     FORMAT(5X,'THE SPECIFIED TIME IS :',G15.5/
     1        5X,'THE MAIN TITLE IS :'/A79/5X,
     2           'IF NO CHANGE IS DESIRED ENTER 1 OR ELSE INPUT TITLE')
       READ (JPLOTD,70) PLTITL
70     FORMAT(A)

       JFILE = 0
       IF (PLTITL(1:1) .EQ. '9') THEN
71       WRITE(KTERMO,*) ' INPUT FILE UNIT TO READ TITLE AND SUB TITLE'
         READ(JPLOTD,*,ERR=71) JFILE
         READ(JFILE,70) PLTITL
         IF (PLTITL(1:1) .NE. '1') MTITLE = PLTITL
         IF (PLTITL(2:2) .EQ. '2') READ(JFILE,70) PLTITL
         GO TO 75
       ENDIF

       IF (PLTITL(1:1) .NE. '1') MTITLE = PLTITL
       PLTITL = ' '
75     CALL GRINIT(KTERMI, KTERMO, MTITLE)

2001   WRITE(KTERMO,*) ' INPUT TIME OF THE RUN [<0 FOR BLANK]'
       READ (JPLOTD,*,ERR=2001) RTIME
       WRITE(KTERMO,*) ' RTIME',RTIME

C      SEE IF THE DEVICE NUMBER IS CORRECT, AND IF SO CHECK IF
C      THE TERMINAL IS MONOCHROME
C
       CALL LIB$GET_SYMBOL(%DESCR('DEV'),%DESCR(DEVNAM),,)
       IF (DEVNAM .NE. 'VR260') THEN
          WRITE(KTERMO,80) DEVNAM
          IFLAGC = 0
          GOTO 90
       ENDIF
80     FORMAT(' COLOR CONTOUR ROUTINE CAN NOT BE USED WITH DEVICE ',A5)

       CALL GKS$INQ_COLOR_FAC(0,IERRST,NUM_COLOR,IFLAGC,
     1                        NUM_INDEX)
C      IERRST IS THE ERROR STATUS; IT MUST BE ZERO
C      NUM_COLOR IS THE NUMBER OF AVAILABLE COLORS
C      IFLAGC IS THE COLOR FLAG; (0: MONOCHROME  1:COLOR)
C      NUM_INDEX IS THE NUMBER OF PREDEFINED INDICES

90     CONTINUE
C      WRITE (KTERMO,100)
100    FORMAT (' INPUT THE VARIABLE TO SET THE DATE AS FOLLOWS:'/
     1      5X,' 1. SET TO BLANKS '/
     2      5X,' 2. USE TODAY''S DATE'/
     3      5X,' 3. DEFINE YOUR OWN CHARACTERS'/                )
```

```
C
      IF (NODTYP .EQ. 1) THEN
         IDATE  = ' '
         ITIME  = ' '
         CALL GR_SET_TIME (IDATE, ITIME)
      ENDIF
C
      IF (NODTYP .EQ. 3) THEN
         WRITE (KTERMO,110)
         READ  (KTERMI,120) IDATE, ITIME
         CALL GR_SET_TIME (IDATE, ITIME)
      ENDIF
110   FORMAT(' INPUT DATE AND TIME')
120   FORMAT(A9, A8)
C
C     INITIALIZE THE MAX/MIN COORDINATES AND VECTOR COMPONENTS
C
      XMIN = 1.E20
      XMAX =-1.E20
      YMIN = 1.E20
      YMAX =-1.E20
      UMIN = 1.E20
      UMAX =-1.E20
      VMIN = 1.E20
      VMAX =-1.E20
C
C     FIND THE SCALE FACTORS FOR X AND Y AXES

      DO  130 INODE = 1, NNODG2
          XMIN  = MIN   (XMIN ,ZX(INODE))
          YMIN  = MIN   (YMIN ,ZY(INODE))
          XMAX  = MAX   (XMAX ,ZX(INODE))
          YMAX  = MAX   (YMAX ,ZY(INODE))
          UMIN  = MIN   (UMIN ,DPENG2(2,INODE))
          VMIN  = MIN   (VMIN ,DPENG2(3,INODE))
          UMAX  = MAX   (UMAX ,DPENG2(2,INODE))
          VMAX  = MAX   (VMAX ,DPENG2(3,INODE))
130   CONTINUE
C
C     SETUP GRDUYMY

      GRDUMY( 1) = NCELA2
      GRDUMY( 2) = NNODG2
      GRDUMY( 3) = NCELG2
      GRDUMY( 4) = NBNDG2
      GRDUMY( 5) = XMIN
      GRDUMY( 6) = XMAX
      GRDUMY( 7) = YMIN
      GRDUMY( 8) = YMAX
      GRDUMY( 9) = UMIN
      GRDUMY(10) = UMAX
      GRDUMY(11) = VMIN
      GRDUMY(12) = VMAX
C
C     SEE IF BLACK BACKGROUND SCREEN IS NEEDED
C
      IF (DEVNAM .EQ. 'VR260' .AND. IBKGRN .EQ. 1) THEN
```

```
        CALL GR_SET_COLOR(1,1,0.,0.,0.)
        CALL GR_SET_COLOR(1,2,1.,1.,1.)
      ENDIF

      CALL SET_ASTC
C
C     DECIDE UPON THE TYPE OF PLOT THAT IS NEEDED
C
1000  WRITE (KTERMO,1010)
1010  FORMAT (' THE FOLLOWING TYPES CAN BE PLOTTED '//
     1       '  1. COLOR CONTOUR'/
     2       '  2. LINE CONTOUR'/
     3       '  3. GRID PLOT'/
     4       '  4. VECTOR PLOT'/
     5       '  5. LINE PLOT'/
     6       '  6. REQUEST VALUES '/
     6       '  7. BLACK AND WHITE CONTOURS '/
     7       '  8. EXIT'/)
      READ (KTERMI,*,ERR=1000) KCTYPE

      GOTO (2000, 3000, 4000, 5000, 6000, 7000, 8000, 9001), KCTYPE
      GOTO 1000

2000  CONTINUE

      INDMIN = INCMIN
      INDMAX = INCMAX
      GRDUMY(26) = RTIME
C
C                 --------------
      PLOT_TYPE = 'COLOR CONTOURS'
C                 --------------
C
2002  WRITE (KTERMO,2010) PLOT_TYPE
2010  FORMAT (' THE FOLLOWING ',A,' CAN BE GENERATED'//
     1       '  1. DENSITY'/
     2       '  2. U VELOCITY COMPONENT'/
     3       '  3. V VELOCITY COMPONENT'/
     4       '  4. TOTAL ENERGY PER UNIT VOLUME'/
     5       '  5. PRESSURE'/
     6       '  6. TEMPERATURE'/
     7       '  7. STAGNATION ENTHALPY'/
     8       '  8. MACH NUMBER'/
     9       '  9. MASS FRACTIONS -- 11 TO 15'/)
      READ (KTERMI,*,ERR=2002) KCONT
C
C     COMPUTE THE MAX/MIN VALUES FOR THE CONTOURS
C
      ZMIN = 1.E20
      ZMAX =-1.E20
      NODMIN = 0
      NODMAX = 0
      DO  2020 INODE = 1, NNODG2
          ZF(INODE) = XYPLOT(KCONT,INODE)
          IF (ZF(INODE) .LT. ZMIN) THEN
              ZMIN   = ZF(INODE)
              NODMIN = INODE
```

```
          ENDIF
          IF (ZF(INODE) .GT. ZMAX) THEN
            ZMAX   = ZF(INODE)
            NODMAX = INODE
          ENDIF
2020    CONTINUE
C
        WRITE(KTERMO,2030) ZMIN, ZMAX, NODMIN, NODMAX
2030    FORMAT(' THE MAX/MIN VALUES OF CONTOURS ARE'/
     1          5X,'ZMIN =',G14.5,10X,'ZMAX =',G14.5/
     2          5X,'NODE OF MIN VALUE =',I5,1X,
     3          5X,'NODE OF MAX VALUE =',I5/)

        WRITE(KTERMO,2040)
2040    FORMAT(' WANT TO DEFINE YOUR OWN EXTREMUM VALUES ?')

        READ (KTERMI,2100) YESNO
        IF (YESNO .EQ. 'z' .OR. YESNO .EQ. 'Z') GOTO 1000

        IF (IFLAGC .EQ. 0 .AND. DEVNAM .NE. 'VR260') THEN
           WRITE(KTERMO,2050)
           GOTO 1000
        ENDIF
2050    FORMAT(' COLOR CONTOUR ROUTINE CAN NOT BE USED WITH MONOCHROME')

        IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') THEN
          CALL GR_REAL('ENTER MIN CONTOUR VALUE',ZMIN)
          CALL GR_REAL('ENTER MAX CONTOUR VALUE',ZMAX)
        ENDIF

2051    WRITE(KTERMO,2060)
2060    FORMAT(' INPUT THE COLOR KEY INDICATOR'/)
        READ (KTERMI,*,ERR=2051) ICOLBL
2061    WRITE (KTERMO,2070)
2070    FORMAT (' INPUT THE MAXIMUM DIFFERENCE OF TOLERABLE INDICES')
        READ (KTERMI,*,ERR=2061) INDDFM

        GRDUMY(13) = ZMIN
        GRDUMY(14) = ZMAX
        GRDUMY(15) = ICOLBL
        GRDUMY(16) = INDDFM
        GRDUMY(27) = INDMIN
        GRDUMY(28) = INDMAX
C
C        IF (JFILE .EQ. 0) THEN
C          WRITE(PLTITL,2080) E1TAX$(KCONT)
C        ENDIF
C2080    FORMAT(' X-AXIS  Y-AXIS           CONTOURS OF ',A8)

        WRITE(PLTITL,2080) E1TAX$(KCONT)
2080    FORMAT(' - -    CONTOURS OF ',A8)

        KNDGR = 23
        KOPT  = 2
        CALL PLXST2(KNDGR)
C
        CALL GK_CONTROL (ZRPLTC, ZRPLTL,ZRPLTG,ZRVECT,INDFIL,
```

```
     1   KNDGR, PLTITL,
     2   MBNDG2,ICELG2,ICELA2,ZX,ZY,ZF,GRDUMY,IBNDG2,KAUXG2,IMARKN,
     3   ICELG2,ICELA2,ZX,ZY,ZF,GRDUMY,CONT,IBNDG2,KAUXG2,IMARKN,
     4   ICELG2,ICELA2,KAUXG2,ZX,ZY,GRDUMY,Z7,Z8,Z9,Z10,
     5   ZX,ZY,PRESG2,TEMPG2,KAUXG2,GRDUMY,ICELG2,ICELA2,IPOINT,IMARKN)
C
      WRITE(KTERMO,2090) PLOT_TYPE
2090  FORMAT(' WANT TO GENERATE MORE ',A,' ?  [Y/N/D] ')
      READ(KTERMI,2100) YESNO
2100  FORMAT(A1)
      IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 2002
      GOTO 1000

3000  CONTINUE
C
C                  -------------
      PLOT_TYPE = 'LINE CONTOURS'
C                  -------------
C
3001  WRITE (KTERMO,2010) PLOT_TYPE
      READ (KTERMI,*,ERR=3001) KCONT
C
C     COMPUTE THE MAX/MIN VALUES FOR THE CONTOURS
C
      ZMIN = 1.E20
      ZMAX =-1.E20
      DO 3010 INODE = 1, NNODG2
          ZF(INODE) = XYPLOT(KCONT,INODE)
          ZMIN      = MIN   (ZMIN ,XYPLOT(KCONT,INODE))
          ZMAX      = MAX   (ZMAX ,XYPLOT(KCONT,INODE))
3010  CONTINUE
C
3011  WRITE(KTERMO,2030) ZMIN, ZMAX
      WRITE (KTERMO,3020)
3020  FORMAT(' INPUT THE NUMBER (NCONT) OF CONTOURS DESIRED:'/5X,
     1  '1. NCONT < 0           : ABS(NCONT) CONTOURS ARE PLOTTED'/5X,
     2  '2. NCONT > 2000        : NCONT-2000 CONTOURS ARE PLOTTED'/5X,
     3  '3. 1000 < NCONT < 2000 : NCONT-2000 CONTOURS ARE PLOTTED'/
     4  5X, 'AUTOMATIC SCALING IS DONE FOR CASES 2 AND 3'             )
      READ (KTERMI,*,ERR=3011) NCONT
C
      NC1    = ABS (NCONT)
         IF (NCONT .LT. 0) THEN
         IDENO  = MAX(1,NC1-1)
         ZSTEP  = (ZMAX-ZMIN)/IDENO
         ZCBASE = ZMIN  - 1.E-6
         ZCSTEP = ZSTEP + 1.E-6
3021     WRITE (KTERMO,3030) ZMIN, ZSTEP
         READ  (KTERMI,*,ERR=3031) ZCBASE
         READ  (KTERMI,*,ERR=3021) ZCSTEP
      ENDIF
3030  FORMAT('  CONTOURS ARE DEFINED BY:'/5X,
     1    'CONTOUR(I)=ZCBASE + (I-1)*ZCSTEP ; I = 1 TO # CONTOURS'/
     2    5X,'DEFAULT ZCBASE AND ZCSTEP',2G14.5                    /
     3    5X,'INPUT ZCBASE AND ZCSTEP [INPUT A TO SKIP]'           )
C
C     MCONTS IS THE ACTUAL NUMBER OF CONTOURS, NLABEL IS NUMBER
```

```
C      OF LABELS ON THE RHS
C
3031   MCONTS = MOD (NC1,1000)
       NC2    = NC1 / 1000
C
       NLABEL = 0
       ICINC  = 0

       IF (NC2 .EQ. 2) THEN
           NLABEL = MIN (10,MCONTS)
           ICINC  = MCONTS / NLABEL
       ENDIF
C
C      FIND THE CONTOUR LEVELS (ZCBASE : BASE CONTOUR LEVEL,
C                               ZCSTEP : CONTOUR INCREMENT )
C
       IF (NCONT .GT. 0) THEN
          CALL GR_SCALE (ZMIN, ZMAX, MCONTS-1, ZCBASE, ZCSTEP)
       ENDIF
C
       WRITE(PLTITL,3040) E1TAX$(KCONT), ZCSTEP
3040   FORMAT(' X-AXIS  Y-AXIS         CONTOURS OF ',A8,
     1        ' INTERVAL = ',G14.3)

       KNDGR = 23
       KOPT  = 2
       CALL PLXST2(KNDGR)

3041   WRITE(KTERMO,3050)
3050   FORMAT('  INPUT SYMBOL NUMBER IF INTERFACE MARKS ARE DESIRED')
       READ (KTERMI,*,ERR=3041) INTERF

       GRDUMY(13) = ZMIN
       GRDUMY(14) = ZMAX
       GRDUMY(17) = MCONTS
       GRDUMY(18) = ZCBASE
       GRDUMY(19) = ZCSTEP
       GRDUMY(20) = NLABEL
       GRDUMY(21) = ICINC
       GRDUMY(22) = INTERF

       CALL GK_CONTROL (ZRPLTL, ZRDUMY,ZRPLTG,ZRDUMY,INDFIL,
     1   KNDGR, PLTITL,
     2   ICELG2,ICELA2,ZX,ZY,ZF,GRDUMY,CONT,IBNDG2,KAUXG2,IMARKN,
     3   DUM1,DUM2,DUM3,DUM4,DUM5,DUM6,DUM7,DUM8,DUM9,DUM10,
     4   ICELG2,ICELA2,KAUXG2,ZX,ZY,GRDUMY,Z7,Z8,Z9,Z10,
     5   DUM1,DUM2,DUM3,DUM4,DUM5,DUM6,DUM7,DUM8,DUM9,DUM10)

       WRITE(KTERMO,2090) PLOT_TYPE
       READ(KTERMI,2100) YESNO
       IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 3000
       GOTO 1000

4000   CONTINUE
C
C                  ----------
       PLOT_TYPE = 'GRID PLOTS'
```

```
C                 ----------
C
      KNDGR = 23
      KOPT  = 2
      ZX1   = XMIN
      ZX2   = XMAX
      ZY1   = YMIN
      ZY2   = YMAX

4001  WRITE(KTERMO,4010)
4010  FORMAT ( 5X, 'INPUT THE PLOT VARIABLES',/
     1         10X, '0.  USE FULL VALUES'/
     2         10X, '1.  SET SCALES OF THE CURVES'/
     2         10X, '2.  USE DEFAULT VALUES'/
     3         10X, ' ==> ',$)
      READ (KTERMI,*,ERR=4001) ITYPE

      IF (ITYPE .EQ. 2)  GOTO 4030
      KNDGR = 22
      IF (ITYPE .EQ. 1) THEN
4011    WRITE(KTERMO,4020)
        READ (KTERMI,*,ERR=4011) ZX1, ZX2, ZY1, ZY2
      ENDIF
4020  FORMAT ( 5X, 'INPUT THE SCALE VALUES XMIN, XMAX, YMIN, YMAX'/
     1         10X, ' ==> ',$)
      CALL GRSSET (ZX1, ZX2, ZY1, ZY2)
C
4030  WRITE(PLTITL,4040)
4040  FORMAT(' X-AXIS  Y-AXIS             GRID PLOT ')

      GRDUMY( 1) = NCELA2
      GRDUMY( 5) = XMIN
      GRDUMY( 6) = XMAX
      GRDUMY( 7) = YMIN
      GRDUMY( 8) = YMAX
      GRDUMY(24) = 0.
      GRDUMY(25) = 0.

      CALL GR_CONTROL (ZRPLTG, KNDGR, PLTITL,
     1          ICELG2,ICELA2,KAUXG2,ZX,ZY,GRDUMY,Z7,Z8,Z9,Z10)

      WRITE(KTERMO,2090) PLOT_TYPE
      READ(KTERMI,2100) YESNO
      IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 4000
      GOTO 1000

5000  CONTINUE
C
C                 -----------
      PLOT_TYPE = 'VECTOR PLOT'
C                 -----------
C
      YESNO     = 'N'
      NNODGR    = NNODG2
5010  CONTINUE
C
      DO 5011 I = 1, NNODG2
```

```
         PRESG2(I)    = DPENG2(2,I)
         TEMPG2(I)    = DPENG2(3,I)
5011   CONTINUE
C
5012   WRITE (KTERMO,5015)
5015   FORMAT (' INPUT THE TYPE OF NODES TO BE PLOTTED AS FOLLOWS:'/
     1    5X,'-2. CORNER NODES OF LEVEL -2 CELLS'/
     2    5X,'-1. CORNER NODES OF LEVEL -1 CELLS'/
     3    5X,' 0. CORNER NODES OF BASE LEVEL CELLS'/
     4    5X,' 1. CORNER NODES OF LEVEL 1 CELLS AND ALL BASE NODES'/
     5    5X,'99. ALL THE NODES'/                                    )
       READ (KTERMI,*,ERR=5012) NODTYP
C
       PLTITL(1:16) = ' X-AXIS  Y-AXIS '
       KNDGR    = 23
       KOPT     = 2
       CALL PLXST2(KNDGR)
C
C      RESET THE MARKED NODES IF PLOTTING MORE THAN ONCE
C
       IF (YESNO .NE. 'N') THEN
          DO  5020 INODE = 1, NNODGR
             IMARKN(INODE) = 0
5020      CONTINUE
          NNODGR   = NNODG2
       ENDIF
C
C      ---------
C      ALL NODES
C      ---------
C
C      SEE IF ALL NODES ARE TO BE USED
C
       IF (NODTYP .EQ. 99) THEN
          MAXNOD = NNODGR
          DO 5030 INODE = 1, NNODGR
             IPOINT(INODE) = INODE
5030      CONTINUE
          GO TO 5110
       ENDIF
C
C      ------------------
C      MULTIPLE GRID NODES
C      ------------------
C
C      CHECK THE NODES AT MULTIPLE GRID LEVELS -1 OR -2
C
       IF (NODTYP .LT. 0) THEN
          IPOWER = ABS(NODTYP)
          IPOWER = 2**IPOWER
C         SET THE STARTING NODE AND INITIALIZE NO. OF ELIGIBLE NODES
          INODE  = 1
          KOUNTE = 0
C         FIRST SCAN X-AXIS AND COLLECT THE APPRORIATE NODES
C         INITIALIZE THE NO. OF NODES TO BE SKIPPED
          KOUNTN = 0
          NBTYPE = 3
```

```
      INTYPE = 4
5040  NCELL  = NEIBG2(NBTYPE,INODE)
5050  IF (NCELL .EQ. 0) THEN
         GOTO 5060
      ELSE
         INODE  = ICELG2(INTYPE,NCELL)
         KX     = KAUXG2(NCELL)
         K5LEVI = IAND(KX,KU000F)
         LEVELI = ISHFT(K5LEVI,-16)
         IF (LEVELI .EQ. 0) THEN
            KOUNTN = KOUNTN + 1
            IF (KOUNTN .EQ. IPOWER) THEN
               KOUNTE         = KOUNTE + 1
               KOUNTN         = 0
               IPOINT(KOUNTE) = INODE
            ENDIF             ! APPROPRIATE NODE FOUND
            GOTO 5040
         ELSE
            NCELL = ICELG2(10,NCELL)
            GOTO 5050
         ENDIF                ! BASE LEVEL CELL
      ENDIF                   ! NEIGHBOUR CELL FOUND
      GOTO 5040
C     NOW MARCH VERTICALLY FROM EACH X-AXIS NODE COLLECTED PREVIOUSLY
5060  DO 5090 KOUNT = 1, KOUNTE
         INODE  = IPOINT(KOUNT)
         NBTYPE = 3
         INTYPE = 8
         NCELL  = NEIBG2(NBTYPE,INODE)
         IF (NCELL .EQ. 0) THEN
            NBTYPE = 4
            INTYPE = 6
         ENDIF
C        INITIALIZE THE NO. OF NODES TO BE SKIPPED
         KOUNTN = 0
5070      NCELL  = NEIBG2(NBTYPE,INODE)
5080      IF (NCELL .EQ. 0) THEN
            GOTO 5090
         ELSE
            INODE  = ICELG2(INTYPE,NCELL)
            KX     = KAUXG2(NCELL)
            K5LEVI = IAND(KX,KU000F)
            LEVELI = ISHFT(K5LEVI,-16)
            IF (LEVELI .EQ. 0) THEN
               KOUNTN = KOUNTN + 1
               IF (KOUNTN .EQ. IPOWER) THEN
                  KOUNTE = KOUNTE + 1
                  KOUNTN = 0
                  IPOINT(KOUNTE) = INODE
               ENDIF             ! APPROPRIATE NODE FOUND
               GOTO 5070
            ELSE
               NCELL = ICELG2(10,NCELL)
               GOTO 5080
            ENDIF                ! BASE LEVEL CELL
         ENDIF                   ! NEIGHBOUR CELL FOUND
         GOTO 5070
```

```
5090     CONTINUE
C
         MAXNOD = KOUNTE
         GO TO 5110
C
      ENDIF                        ! -1, -2 LEVELS
C
C     ----------------------
C     BASE/HIGHER LEVEL CELLS
C     ----------------------
C
C     CHECK THE NODES AT LEVEL 0  AND 1 ONLY
C
      KOUNT = 0

      DO 5100 ICELL = 1, NCELG2

C        SET THE POINTERS FOR THIS CELL

         KC     = ICELG2(1,ICELL)
         KSW    = ICELG2(2,ICELL)
         KS     = ICELG2(3,ICELL)
         KSE    = ICELG2(4,ICELL)
         KE     = ICELG2(5,ICELL)
         KNE    = ICELG2(6,ICELL)
         KN     = ICELG2(7,ICELL)
         KNW    = ICELG2(8,ICELL)
         KW     = ICELG2(9,ICELL)
         KX     = KAUXG2(ICELL)
         K5LEVI = IAND(KX,KUOOOF)
         LEVELI = ISHFT(K5LEVI,-16)
C
C        FOR LEVEL ZERO ONLY USE CORNER CELLS IF TYPE 1
C        MARK THE NODES WHICH ARE DONE WITH IMARKN(NODE)=-1
C
         IF (LEVELI .EQ. 0) THEN
            IF (IMARKN(KSW) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KSW)   = -1
               IPOINT(KOUNT) = KSW
            ENDIF
C
            IF (IMARKN(KSE) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KSE)   = -1
               IPOINT(KOUNT) = KSE
            ENDIF
C
            IF (IMARKN(KNE) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KNE)   = -1
               IPOINT(KOUNT) = KNE
            ENDIF
C
            IF (IMARKN(KNW) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KNW)   = -1
```

```
            IPOINT(KOUNT) = KNW
          ENDIF
C
        ENDIF
C
C       FOR LEVEL ONE ONLY USE CORNER CELLS IF TYPE 2
C       AND FOR LEVEL ZERO USE ALL THE NODES
C
        IF (NODTYP .EQ. 1) THEN
C
C          FIRST CHECK LEVEL 1 CELLS
C
           IF (LEVELI .EQ. 1) THEN
C
             IF (IMARKN(KSW) .NE. -1) THEN
                KOUNT         = KOUNT + 1
                IMARKN(KSW)   = -1
                IPOINT(KOUNT) = KSW
             ENDIF
C
             IF (IMARKN(KSE) .NE. -1) THEN
                KOUNT         = KOUNT + 1
                IMARKN(KSE)   = -1
                IPOINT(KOUNT) = KSE
             ENDIF
C
             IF (IMARKN(KNE) .NE. -1) THEN
                KOUNT         = KOUNT + 1
                IMARKN(KNE)   = -1
                IPOINT(KOUNT) = KNE
             ENDIF
C
             IF (IMARKN(KNW) .NE. -1) THEN
                KOUNT         = KOUNT + 1
                IMARKN(KNW)   = -1
                IPOINT(KOUNT) = KNW
             ENDIF
C
C          NOW CHECK LEVEL 0 CELLS
C
           ELSE IF (LEVELI .EQ. 0) THEN
C
              IF (KC .EQ. 0) THEN
                KOUNT        = KOUNT + 1
                NNODGR       = NNODGR +1
                KC           = NNODGR
                IPOINT(KOUNT)= NNODGR
                ZX(NNODGR)   = 0.25*(ZX(KSW)+ZX(KSE)+ZX(KNE)+ZX(KNW))
                ZY(NNODGR)   = 0.25*(ZY(KSW)+ZY(KSE)+ZY(KNE)+ZY(KNW))
                PRESG2(NNODGR)= 0.25*(PRESG2(KSW) + PRESG2(KSE) +
     1                                PRESG2(KNE) + PRESG2(KNW) )
                TEMPG2(NNODGR)= 0.25*(TEMPG2(KSW) + TEMPG2(KSE) +
     1                                TEMPG2(KNE) + TEMPG2(KNW) )
                IMARKN(KC)   = -1
                ICELG2(1,ICELL) = NNODGR
              ELSE
                IF (IMARKN(KC) .NE. -1) THEN
```

```
            KOUNT          = KOUNT + 1
            IMARKN(KC)     = -1
            IPOINT(KOUNT) = KC
          ENDIF
        ENDIF              ! IF KC = 0
C
        IF (KS .EQ. 0) THEN
            KOUNT           = KOUNT + 1
            NNODGR          = NNODGR +1
            KS              = NNODGR
            IPOINT(KOUNT)   = NNODGR
            ZX(NNODGR)      = 0.5*(ZX(KSW)+ZX(KSE))
            ZY(NNODGR)      = 0.5*(ZY(KSW)+ZY(KSE))
            PRESG2(NNODGR)  = 0.5*(PRESG2(KSW)+PRESG2(KSE))
            TEMPG2(NNODGR)  = 0.5*(TEMPG2(KSW)+TEMPG2(KSE))
            ICELG2(3,ICELL) = NNODGR
            IMARKN(KS)      = -1
            NCELL           = NEIBG2(2,KSW)
            IF (NCELL .NE. 0) ICELG2(7,NCELL) = NNODGR
        ELSE
            IF (IMARKN(KS) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KS)    = -1
               IPOINT(KOUNT) = KS
            ENDIF
        ENDIF              ! IF KS = 0
C
        IF (KE .EQ. 0) THEN
            KOUNT           = KOUNT + 1
            NNODGR          = NNODGR +1
            KE              = NNODGR
            IPOINT(KOUNT)   = NNODGR
            ZX(NNODGR)      = 0.5*(ZX(KSE)+ZX(KNE))
            ZY(NNODGR)      = 0.5*(ZY(KSE)+ZY(KNE))
            PRESG2(NNODGR)  = 0.5*(PRESG2(KSE)+PRESG2(KNE))
            TEMPG2(NNODGR)  = 0.5*(TEMPG2(KSE)+TEMPG2(KNE))
            ICELG2(5,ICELL) = NNODGR
            IMARKN(KE)      = -1
            NCELL           = NEIBG2(3,KSE)
            IF (NCELL .NE. 0) ICELG2(9,NCELL) = NNODGR
        ELSE
            IF (IMARKN(KE) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KE)    = -1
               IPOINT(KOUNT) = KE
            ENDIF
        ENDIF              ! IF KE = 0
C
        IF (KN .EQ. 0) THEN
            KOUNT           = KOUNT + 1
            NNODGR          = NNODGR +1
            KN              = NNODGR
            IPOINT(KOUNT)   = NNODGR
            ZX(NNODGR)      = 0.5*(ZX(KNW)+ZX(KNE))
            ZY(NNODGR)      = 0.5*(ZY(KNW)+ZY(KNE))
            PRESG2(NNODGR)  = 0.5*(PRESG2(KNW)+PRESG2(KNE))
            TEMPG2(NNODGR)  = 0.5*(TEMPG2(KNW)+TEMPG2(KNE))
```

```
                    ICELG2(7,ICELL) = NNODGR
                    IMARKN(KN)      = -1
                    NCELL           = NEIBG2(4,KNE)
                    IF (NCELL .NE. 0) ICELG2(3,NCELL) = NNODGR
                ELSE
                    IF (IMARKN(KN) .NE. -1) THEN
                       KOUNT         = KOUNT + 1
                       IMARKN(KN)    = -1
                       IPOINT(KOUNT) = KN
                    ENDIF
                ENDIF              ! IF KN = 0
C
                IF (KW .EQ. 0) THEN
                    KOUNT           = KOUNT + 1
                    NNODGR          = NNODGR +1
                    KW              = NNODGR
                    IPOINT(KOUNT)   = NNODGR
                    ZX(NNODGR)      = 0.5*(ZX(KSW)+ZX(KNW))
                    ZY(NNODGR)      = 0.5*(ZY(KSW)+ZY(KNW))
                    PRESG2(NNODGR)  = 0.5*(PRESG2(KSW)+PRESG2(KNW))
                    TEMPG2(NNODGR)  = 0.5*(TEMPG2(KSW)+TEMPG2(KNW))
                    ICELG2(9,ICELL) = NNODGR
                    IMARKN(KW)      = -1
                    NCELL           = NEIBG2(1,KNW)
                    IF (NCELL .NE. 0) ICELG2(5,NCELL) = NNODGR
                ELSE
                    IF (IMARKN(KW) .NE. -1) THEN
                       KOUNT         = KOUNT + 1
                       IMARKN(KW)    = -1
                       IPOINT(KOUNT) = KW
                    ENDIF
                ENDIF              ! IF KW = 0
C
            ENDIF                  ! IF BASE LEVEL
          ENDIF                    ! IF NODTYP = 1
C
5100    CONTINUE
C
        MAXNOD = KOUNT
C
5110    GRDUMY(23) = MAXNOD
C
        CALL GR_CONTROL (ZRVECT,KNDGR,PLTITL,
     &  ZX,ZY,PRESG2,TEMPG2,KAUXG2,GRDUMY,ICELG2,ICELA2,IPOINT,IMARKN)

        WRITE(KTERMO,2090) PLOT_TYPE
        READ(KTERMI,2100) YESNO
        IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 5010
        GOTO 1000
C
6000    CONTINUE
C
C                   ----------
        PLOT_TYPE = 'LINE PLOTS'
C                   ----------
C
        PLTITL    = ' '
```

```
      SCALING   = 1.
CDEBUG
6010  WRITE(KTERMO,6020)
6020  FORMAT(' INPUT NODE FOR WHICH NB IS DESIRED')
      READ(KTERMI,*,ERR=6010) INNB
CDEBUG
      IF (INNB .LT. 0) THEN
6021     WRITE(KTERMO,*) ' INPUT SCALING FACTOR'
         READ(KTERMI,*,ERR=6021) SCALING
         GOTO 6050
      ENDIF
CDEBUG
      IF (INNB .NE. 0) THEN
         WRITE(KTERMO,6030) (NEIBG2(IK,INNB),IK=1,4)
         WRITE(KTERMO,6040) (ICELG2(IK,NEIBG2(2,INNB)),IK=2,8,2)
         GOTO 6010
      ENDIF
6030  FORMAT(2X,'NEIGHBOUR CELLS               ',2X,4I5)
6040  FORMAT(2X,'NEIGHBOUR NODES OF SECOND CELL',2X,4I5)
CDEBUG

6050  CONTINUE
      WRITE (KTERMO,2010) PLOT_TYPE
6051  WRITE(KTERMO,6060)
6060  FORMAT(' INPUT NEGATIVE VALUE TO PLOT CURVILINEAR DISTANCE'
     1       /' ===> ',$)
      READ (KTERMI,*,ERR=6051) KCONT
C
6061  WRITE(KTERMO,6070)
6070  FORMAT(' INPUT THE STARTING POINT'/
     1        ' INPUT NEGATIVE VALUE TO MARCH TO NORTH'/' ===> ',$)
      READ (KTERMI,*,ERR=6061) ISTART
      INODE = ABS(ISTART)
C
C     CHECK IF THE NODE IS A BOUNDARY NODE AT THE APPROPRIATE SIDE
C
      IF (ISTART .GT. 0) THEN
         IEDGE = 9
         DO 6080 INBND = 1, NBNDG2
           IF (IBNDG2(4,INBND) .EQ. 9) THEN
             IF (IBNDG2(1,INBND) .EQ. INODE) GOTO 6100
           ENDIF
6080       CONTINUE
      ENDIF
C
      IF (ISTART .LT. 0) THEN
         IEDGE = 3
         DO 6090 INBND = 1, NBNDG2
           IF (IBNDG2(4,INBND) .EQ. 3) THEN
             IF (IBNDG2(1,INBND) .EQ. INODE) GOTO 6100
           ENDIF
6090       CONTINUE
      ENDIF
      IF (INODE .EQ. 1) GOTO 6100
C
C     WARNING CONDITION
C
```

```
      ZER1 = INODE
      ZER2 = IEDGE
      CALL WARNIN (45,'P2LINE','INODE ',ZER1,'IEDGE ',ZER2,JPRINT,
     1           'THE NODE IS NOT ON THE CORRECT BOUNDARY')

6100  IF (ISTART .GT. 0) THEN
        PLTITL(1:8) = 'X       '
        NB1 = NEIBG2(2,INODE)
        NB2 = NEIBG2(3,INODE)
        NBTYPE = 0
        IF (NB1 .NE. 0) THEN
          NBTYPE = 2
          INTYPE = 6
        ELSEIF (NB2 .NE. 0) THEN
          NBTYPE = 3
          INTYPE = 4
        ENDIF
        IF (NB1 .NE. 0 .AND. NB2 .NE. 0) THEN
6101      WRITE(KTERMO,6110)
          READ(KTERMI,*,ERR=6101) KOPT
          IF (KOPT .EQ. 2) THEN
            NBTYPE = 3
            INTYPE = 4
          ENDIF
        ENDIF
      ENDIF

6110  FORMAT(1X,'INPUT ONE OF THE FOLLOWING'/
     1       5X,'1. Lower horizontal surface'/
     2       5X,'2. Upper horizontal surface'/' ===> ',$)
C
      IF (ISTART .LT. 0) THEN
        PLTITL(1:8) = 'Y       '
        NB1 = NEIBG2(3,INODE)
        NB2 = NEIBG2(4,INODE)
        NBTYPE = 0
        IF (NB1 .NE. 0) THEN
          NBTYPE = 3
          INTYPE = 8
        ELSEIF (NB2 .NE. 0) THEN
          NBTYPE = 4
          INTYPE = 6
        ENDIF
      ENDIF
C
C     ERROR CONDITION
C
      IF (NBTYPE .EQ. 0) THEN
        ZER1 = ISTART
        ZER2 = NBTYPE
        CALL ERRORM (46,'P2LINE','ISTART',ZER1,'NBTYPE',ZER2,JPRINT,
     1           'ERROR IN NEIGHBOUR CELLS OF STARTING POINT')
      ENDIF
C
C     NOW MARCH IN THE APPROPRIATE DIRECTION
C
      KOUNT = 0
```

```
      IF (KCONT .LT. 0) THEN
        KCONT = ABS(KCONT)
        XPREV = ZX(INODE)
        YPREV = ZY(INODE)
        SMIN  = SQRT(XPREV**2+YPREV**2)
        SSUM  = 0.
6120    KOUNT         = KOUNT + 1
        TEMPG2(KOUNT) = XYPLOT(KCONT,INODE)
        XHERE         = ZX(INODE)
        YHERE         = ZY(INODE)
        SHERE         = SQRT( (XHERE-XPREV)**2 +(YHERE-YPREV)**2 )
        SSUM          = SSUM + SHERE
        XPREV         = XHERE
        YPREV         = YHERE
        PRESG2(KOUNT) = SSUM
        NBNEXT        = NEIBG2(NBTYPE,INODE)
        IF (NBNEXT .EQ. 0) GOTO 6150
        INODE         = ICELG2(INTYPE,NBNEXT)
        GO TO 6120
      ENDIF

      IF (ISTART .GT. 0) THEN
6130    KOUNT         = KOUNT + 1
        TEMPG2(KOUNT) = XYPLOT(KCONT,INODE)*SCALING
        PRESG2(KOUNT) = ZX(INODE)
        NBNEXT        = NEIBG2(NBTYPE,INODE)
        IF (NBNEXT .EQ. 0) GOTO 6150
        INODE         = ICELG2(INTYPE,NBNEXT)
        GO TO 6130
      ENDIF
C
      IF (ISTART .LT. 0) THEN
6140    KOUNT     = KOUNT + 1
        TEMPG2(KOUNT) = XYPLOT(KCONT,INODE)
        PRESG2(KOUNT) = ZY(INODE)
        NBNEXT    = NEIBG2(NBTYPE,INODE)
        IF (NBNEXT .EQ. 0) GOTO 6150
        INODE     = ICELG2(INTYPE,NBNEXT)
        GO TO 6140
      ENDIF

6150  PLTITL(9:16) = E1TAX$(KCONT)
      KNDGR    = 21
      KOPT     = 2
      CALL PLXSET(KOPT,KNDGR)
      NLINE    = 1
      KKOPT(1) = KOPT
      KN$(1)   = KOUNT

      CALL GR_LINE(KKOPT,NLINE,PLTITL,KNDGR,PRESG2,TEMPG2,KN$)

      WRITE(KTERMO,2090) PLOT_TYPE
      READ(KTERMI,2100) YESNO
      IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 6050

C     WRITE OUTPUT DATA
      IF (YESNO .EQ. 'd' .OR. YESNO .EQ. 'D') THEN
```

```
        ISTRING = ' '
        WRITE(KTERMO,*) ' INPUT THE FILE NAME FOR DATA, OR TYPE QUIT'
        READ (KTERMI,70) ISTRING
        IF (ISTRING(1:4).EQ.'QUIT' .OR. ISTRING(1:4).EQ.'quit')
     1      GOTO 6050
        OPEN (UNIT=58, FILE=ISTRING, STATUS='NEW',FORM='FORMATTED')
        WRITE(KTERMO,6170) ISTRING
        WRITE(58,*) KOUNT
        DO 6160 IK = 1, KOUNT
          WRITE(58,*) PRESG2(IK), TEMPG2(IK)
6160    CONTINUE
        GOTO 6050
      ENDIF
6170  FORMAT (5X,'WRITING OUTPUT ON ',A)

      GOTO 1000

7000  CONTINUE
C
C                 ------
      PLOT_TYPE = 'VALUES'
C                 ------
C
7010  WRITE(KTERMO,7020)
7020  FORMAT(' THE FOLLOWING VALUES CAN BE REQUESTED'/
     1     5X,'1. CPU TIME'/
     2     5X,'2. FCTRTI'/
     3     5X,'3. NEQNFL'/
     4     5X,'4. NEQSCH'/
     5     5X,'5. YNRTCH'/
     6     5X,'6. TRIGCH'/)

      READ(KTERMI,*,ERR=7010) INNB
      IF (INNB .EQ. 1) THEN
         WRITE(KTERMO,*) ' CPU TIME ',ZCUM
      ELSEIF (INNB .EQ. 2) THEN
         WRITE(KTERMO,*) ' FCTRTI ',FCTRTI
      ELSEIF (INNB .EQ. 3) THEN
         WRITE(KTERMO,*) ' NEQNFL ',NEQNFL
      ELSEIF (INNB .EQ. 4) THEN
         WRITE(KTERMO,*) ' NEQSCH ',NEQSCH
      ELSEIF (INNB .EQ. 5) THEN
         WRITE(KTERMO,*) ' YNRTCH ',YNRTCH
      ELSEIF (INNB .EQ. 6) THEN
         WRITE(KTERMO,*) ' TRIGCH ',TRIGCH
      ENDIF

      WRITE(KTERMO,2090) PLOT_TYPE
      READ(KTERMI,2100) YESNO
      IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 7010

      GOTO 1000

8000  CONTINUE
C
C                 ------------
      PLOT_TYPE = 'B&W CONTOURS'
```

```
C                 ------------
C
      INDMIN = 33
      INDMAX = 43

8010  WRITE(KTERMO,8020)
8020  FORMAT(' INPUT THE BACKGROUND COLOR'/
     1    5X,'1. BLACK'/
     2    5X,'2. WHITE')
      READ(KTERMI,*,ERR=8010) IBKGRN

      IF (DEVNAM .EQ. 'VR260' .AND. IBKGRN .EQ. 1) THEN
        CALL GR_SET_COLOR(1,1,0.,0.,0.)
        CALL GR_SET_COLOR(1,2,1.,1.,1.)
      ELSE
        CALL GR_SET_COLOR(1,1,1.,1.,1.)
        CALL GR_SET_COLOR(1,2,0.,0.,0.)
      ENDIF

      GOTO 2002

9001  STOP ' THE END'
      END


      SUBROUTINE PLXST2 (INDGR)
C
      SAVE XMIN, XMAX, YMIN, YMAX
C
C**********************************************************************
C
C     THIS SUBROUTINE SETS THE SCALES OF THE PLOTS IN THE COMMON
C     BLOCKS OF THE GRAFIC ROUTINES.
C
C**********************************************************************
C
C     SET THE DEFAULT VALUES
C
      KTERMI = 5
      KTERMO = 6
C
15    WRITE(KTERMO,20)
20    FORMAT ( 5X, 'INPUT THE PLOT VARIABLES',/
     1         10X, '-1.  USE PREVIOUS SCALE VALUES'/
     1         10X, ' 0.  USE DEFAULT VALUES'/
     3         10X, ' 1.  AUTOMATIC SCALES'/
     5         10X, ' 2.  SAME STEP SIZE ON BOTH AXES'/
     5         10X, ' 4.  DRAW AXES'/
     5         10X, ' 8.  DRAW BACKGROUND GRID'/
     6         10X, ' ==> ',$)
      READ (KTERMI,*,ERR=15) ITYPE

      IF (ITYPE .EQ. 0) RETURN
      IF (ITYPE .LT. 0) THEN
        CALL GRSSET (XMIN, XMAX, YMIN, YMAX)
        RETURN
      ENDIF
```

```
      INDGR_= ITYPE + 16
      IF (IAND (ITYPE,'00000001'X) .EQ. 0) THEN
31      WRITE(KTERMO,40)
        READ (KTERMI,* ,ERR=31) XMIN, XMAX, YMIN, YMAX
        CALL GRSSET (XMIN, XMAX, YMIN, YMAX)
        WRITE(KTERMO,50) XMIN, XMAX, YMIN, YMAX
      ENDIF
40      FORMAT ( 5X, 'INPUT THE SCALE VALUES XMIN, XMAX, YMIN, YMAX'/
     1           10X, ' ==> ',$)
50      FORMAT(10X,'XMIN = ', G14.5, 10X, 'XMAX = ',G14.5/
     1         10X,'YMIN = ', G14.5, 10X, 'YMAX = ',G14.5)

      RETURN
      END
```

P2GRID

```
       PROGRAM P2GRID

       INCLUDE '[PERVAIZ.TWODO.INC] PRECIS.INC/LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] PARMV2.INC/LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] A2COMN.INC /LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] G2COMN.INC /LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] CHCOMN.INC /LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] E2COMN.INC /LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] FLCOMN.INC /LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] HEXCOD.INC     '
       INCLUDE '[PERVAIZ.TWODO.INC] IOCOMN.INC /LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] PRCOMN.INC /LIST'
       INCLUDE '[PERVAIZ.TWODO.INC] TICOMN.INC /LIST'

       DIMENSION NCELGR(0:MLVLG2), ICELTT(0:MLVLG2,MCELG2)
       REAL*4    GRDUMY(30), ALIMITS(6)
       CHARACTER PLTITL*96 , ISTRING*80, YESNO*1, IDATE*9, ITIME*8
       DIMENSION ZX(MNODG2), ZY(MNODG2)
       EXTERNAL ZRPLTG

C**********************************************************************
C
C      THIS PROGRAM READS ALL THE INFORMATION ABOUT THE POINTER SYSTEM
C      AND ALL THE OTHER ARRAYS FROM UNIT 'JPNTRE', FROM A RUN CASE FROM
C      FILE JPNTRE.DAT.  THIS PROGRAM THEN MAKES THE GRID PLOT FOR THE
C      TWO-DIMENSIONAL CASE FOR A GIVEN LEVEL OF CELLS.
C
C**********************************************************************

       JTERMI   = 5
       JTERMO   = 6
       MTITLE   = ' '
       PLTITL = ' '
       JPNTRE = 28
C
C      READ THE POINTER SYSTEM INFORMATION
C
```

```
      WRITE(JTERMO,*) ' READING FROM UNFORMATTED PLOTTING FILE'
      OPEN (UNIT=JPNTRE, FILE='JPNTRE.DAT', STATUS='OLD',
     1       FORM='UNFORMATTED', READONLY)
      CALL PSREDU
C
C     ONLY THE GRIDS FOR THE CEWIC CELLS WILL BE PRODUCED, ALTHOUGH
C     ANY PARTICULAR LEVEL OF CELLS CAN BE PRODUCED
C     CLASSIFY THE CELLS ACCORDING TO THEIR LEVEL
C
      DO 10 IN = 1, NNODG2
        ZX(IN) = GEOMG2(1,IN)
        ZY(IN) = GEOMG2(2,IN)
10    CONTINUE

      DO 20 ILEVEL = 0, MLVLG2
        NCELGR(ILEVEL) = 0
20    CONTINUE
C
      DO 30 JCELL = 1, NCELA2

C       FIND THE ACTUAL CELL NUMBER AND HENCE THE LEVEL AND STORE IT

        ICELL               = ICELA2(JCELL)
        KX                  = KAUXG2(ICELL)
        K5LEVI              = IAND(KX,KU000F)
        LEVELI              = ISHFT(K5LEVI,-16)
        NCELL               = NCELGR(LEVELI) + 1
        NCELGR(LEVELI)      = NCELL
        ICELTT(LEVELI,NCELL) = ICELL

30    CONTINUE
C
C     INITIALIZE THE GRAPHICS ROUTINES
C
      WRITE(JTERMO,40) MTITLE
40    FORMAT(5X,'THE MAIN TITLE IS :'/A79/5X,
     1       'IF NO CHANGE IS DESIRED ENTER 1 OR ELSE INPUT TITLE')
      READ (JTERMI,50) PLTITL
50    FORMAT(A)
      IF (PLTITL(1:1).NE.'1') MTITLE = PLTITL
      PLTITL = ' '

      CALL GR_INIT (JTERMI, JTERMO, MTITLE)
C
C     FIND THE LEVEL OF CELLS TO BE PRINTED
C
60    WRITE (JTERMO,70)
70    FORMAT(/5X,'INPUT THE LEVEL OF CELLS TO BE PLOTTED ?'
     1       /5X,'INPUT 99 IF ALL LEVELS ARE DESIRED')
      WRITE(JTERMO,80)
80    FORMAT(5X,'===> ',$)
      READ(JTERMI,*) LEVELI

      WRITE (JTERMO,90)
      READ (JTERMI,*) NODTYP
90    FORMAT (' INPUT THE VARIABLE TO SET THE DATE AS FOLLOWS:'/
     1    5X,' 1. SET TO BLANKS '/
```

```
     2     5X,' 2. USE TODAY''S DATE'/
     3     5X,' 3. DEFINE YOUR OWN CHARACTERS'/                )
C
       IF (NODTYP .EQ. 1) THEN
          IDATE  = ' '
          ITIME  = ' '
          CALL GR_SET_TIME (IDATE, ITIME)
       ENDIF
C
       IF (NODTYP .EQ. 3) THEN
          WRITE (JTERMO,100)
100    FORMAT(' INPUT DATE AND TIME')
          READ  (JTERMI,110) IDATE, ITIME
110    FORMAT(A9, A8)
          CALL GR_SET_TIME (IDATE, ITIME)
       ENDIF
C
C      INITIALIZE THE MAX/MIN COORDINATES
C
       XMIN = 1.E20
       YMIN = 1.E20
       XMAX =-1.E20
       YMAX =-1.E20

       IF (LEVELI .NE. 99) THEN

C         CHECK FOR SPECIFICATION ERROR IN LEVELI

          IF (LEVELI .LT. 0 .OR. LEVELI .GT. MLVLG2) THEN
            ZER1 = LEVELI
            ZER2 = MLVLG2
            CALL ERRORM (22,'P2GRID','LEVELI',ZER1,'MLVLG2',ZER2,
     1       JPRINT,'NUMBER OF LEVELS IS WRONG')
            GOTO 60
          ENDIF

C         LOOP THROUGH ALL FOUR CORNERS OF THE CELLS AT THIS LEVEL
C         AND COLLECT MAX/MIN INFORMATION AND SET DISTANCE ARRAYS

          GRDUMY(1) = NCELGR(LEVELI)
          DO 130 JCELL = 1, NCELGR(LEVELI)

C            FIND THE ACTUAL CELL NUMBER

             ICELL         = ICELTT(LEVELI,JCELL)
             ICELA2(JCELL) = ICELL

             DO 120 ICORN = 2, 8, 2
               INODE  = ICELG2(ICORN,ICELL)
               ZXNODE = ZX(INODE)
               ZYNODE = ZY(INODE)
               XMIN   = MIN   (XMIN ,ZXNODE)
               YMIN   = MIN   (YMIN ,ZYNODE)
               XMAX   = MAX   (XMAX ,ZXNODE)
               YMAX   = MAX   (YMAX ,ZYNODE)
120          CONTINUE
```

```
130       CONTINUE
          -
        ELSE

C          LOOP THROUGH ALL FOUR CORNERS OF THE CEWIC CELLS AND
C          COLLECT MAX/MIN INFORMATION AND SET DISTANCE ARRAYS

           GRDUMY(1) = NCELA2
           DO 150 JCELL = 1, NCELA2

C             FIND THE ACTUAL CELL NUMBER

              ICELL         = ICELA2(JCELL)
              ICELA2(JCELL) = ICELL

              DO 140 ICORN = 2, 8, 2
                INODE  = ICELG2(ICORN,ICELL)
                ZXNODE = ZX(INODE)
                ZYNODE = ZY(INODE)
                XMIN   = MIN   (XMIN ,ZXNODE)
                YMIN   = MIN   (YMIN ,ZYNODE)
                XMAX   = MAX   (XMAX ,ZXNODE)
                YMAX   = MAX   (YMAX ,ZYNODE)
140           CONTINUE

150        CONTINUE

        ENDIF

        WRITE (JTERMO,160) XMIN, XMAX, YMIN, YMAX
160     FORMAT (5X,'XMIN = ',G14.5, 5X, 'XMAX = ',G14.5/
     1          5X,'YMIN = ',G14.5, 5X, 'YMAX = ',G14.5)

        KNDGR = 23
        KOPT  = 2
        ZX1   = XMIN
        ZX2   = XMAX
        ZY1   = YMIN
        ZY2   = YMAX
C
        WRITE(JTERMO,170)
170     FORMAT ( 5X, 'INPUT THE PLOT VARIABLES',/
     1           10X, '0.  USE FULL VALUES'/
     2           10X, '1.  SET SCALES OF THE CURVES'/
     2           10X, '2.  USE DEFAULT VALUES'/
     3           10X, ' ==> ',$)

        READ (JTERMI,*) ITYPE
        IF (ITYPE .EQ. 2)  GOTO 190
        KNDGR = 22
        IF (ITYPE .EQ. 1) THEN
          WRITE(JTERMO,180)
          READ (JTERMI,* ) ZX1, ZX2, ZY1, ZY2
        ENDIF
180     FORMAT ( 5X, 'INPUT THE SCALE VALUES XMIN, XMAX, YMIN, YMAX'/
     1           10X, ' ==> ',$)
        CALL GRSSET (ZX1, ZX2, ZY1, ZY2)
```

```
C
190     WRITE(PLTITL,200)
200     FORMAT(' X-AXIS  Y-AXIS   GRID PLOT ')

        GRDUMY( 5) = XMIN
        GRDUMY( 6) = XMAX
        GRDUMY( 7) = YMIN
        GRDUMY( 8) = YMAX
        GRDUMY(24) = 0.
        GRDUMY(25) = 0.

        CALL GR_CONTROL (ZRPLTG, KNDGR, PLTITL,
     1     ICELG2,ICELA2,KAUXG2,ZX,ZY,GRDUMY,Z7,Z8,Z9,Z10)

        WRITE(JTERMO,*) ' WANT TO PLOT MORE ?  [Y/N] '

        READ(JTERMI,210) YESNO
210     FORMAT(A1)
        IF (YESNO .EQ. 'y' .OR. YESNO .EQ. 'Y') GOTO 60
C
        END
```

P2ITER

```
        PROGRAM P2ITER

        PARAMETER (MITER = 1000)

        DIMENSION ZERROR(MITER), ZITER(MITER), XYPLOT(3,MITER),
     1            NNOPT(3), N$(3), E1TAX$(3)

        CHARACTER PLTITL*96, YESNO*1, E1TAX$*8, IDATE*9, ITIME*8,
     1            MTITLE*80
        DATA E1TAX$/'ABSOLUTE' ,  'MAXIMUM ' ,  '  RMS    ' /

C*********************************************************************
C
C       THIS PROGRAM GENERATES THE ITERATION PLOTS
C
C*********************************************************************

        JTERMI = 5
        JTERMO = 6
        MTITLE = ' '
        JPRINT = 7
        JHISTO = 8
C
C       READ THE POINTER SYSTEM INFORMATION
C
        OPEN (UNIT=JHISTO, FILE='JHISTO.DAT', STATUS='OLD')

        READ (JHISTO,1000) MTITLE
        NITER = 0
10      READ (JHISTO,1100,END=20) NITRE2, IP    , KONVE2, KEQNE2,
     1                            ERROR1, ERROR2, ERROR3, TIME
```

```
      NITER        = NITER + 1
      ZITER(NITER) = NITER
C
C     NOW SET THESE VALUES IN THE XYPLOT ARRAY
C
      XYPLOT( 1,NITER) = ERROR1
      XYPLOT( 2,NITER) = ERROR2
      XYPLOT( 3,NITER) = ERROR3
C
      GOTO 10
C
20    CONTINUE

C     INITIALIZE THE MAX/MIN COORDINATES
C
      XMIN = 0
      XMAX = NITER
C
C     INITIALIZE THE GRAPHICS ROUTINES
C
      WRITE(JTERMO,1200) MTITLE
      READ (JTERMI,1300) PLTITL
      IF (PLTITL(1:1) .NE. '1') MTITLE = PLTITL
      PLTITL = ' '

      CALL GRINIT(JTERMI, JTERMO, MTITLE)

      WRITE (JTERMO,1400)
      READ (JTERMI,*) IYPLOT
C
      IF (IYPLOT .EQ. 1) THEN
         IDATE  = ' '
         ITIME  = ' '
         CALL GR_SET_TIME (IDATE, ITIME)
      ENDIF
C
      IF (IYPLOT .EQ. 3) THEN
         WRITE (JTERMO,1500)
         READ  (JTERMI,1600) IDATE, ITIME
         CALL GR_SET_TIME (IDATE, ITIME)
      ENDIF

30    WRITE (JTERMO,1700)
      READ (JTERMI,*) IYPLOT
      IYPLOT = ABS(IYPLOT)
C
C     FIND THE SCALE FACTORS FOR X AND Y AXES

      YMIN = 1.E20
      YMAX =-1.E20
      DO 40 INODE = 1, NITER
          ZERROR(INODE) = XYPLOT(IYPLOT,INODE)
          YNODE         = ZERROR(INODE)
          YMIN          = MIN   (YMIN ,YNODE)
          YMAX          = MAX   (YMAX ,YNODE)
40    CONTINUE
```

```
        WRITE (JTERMO,1800) XMIN, XMAX, YMIN, YMAX

        PLTITL(1:10) = 'ITERATION-'
        PLTITL(11:19) = E1TAX$(IYPLOT) // '-'
        INDGR    = 21
        NOPT     = 2
        CALL PLXSET(NOPT,INDGR)
        NLINE    = 1
        NNOPT(1) = NOPT
        N$(1)    = NITER

        CALL GR_LINE(NNOPT,NLINE,PLTITL,INDGR,ZITER,ZERROR,N$)

        WRITE(JTERMO,*) ' WANT TO PLOT MORE ?  [Y/N] '
        READ(JTERMI,1900) YESNO
        IF (YESNO .EQ. 'n' .OR. YESNO .EQ. 'N') STOP

        GOTO 30
C
C       -----------------
C       FORMAT STATEMENTS
C       -----------------
C

1000    FORMAT(A80)
1100    FORMAT(2I5,1X,I2,1X,I2,2X,4G15.5)
1200    FORMAT(5X,'THE MAIN TITLE IS :'/A80/5X,
     1         'IF NO CHANGE IS DESIRED ENTER 1 OR ELSE INPUT TITLE')
1300    FORMAT(A)
1400    FORMAT (' INPUT THE VARIABLE TO SET THE DATE AS FOLLOWS:'/
     1     5X,' 1. SET TO BLANKS '/
     2     5X,' 2. USE TODAY''S DATE'/
     3     5X,' 3. DEFINE YOUR OWN CHARACTERS'/                  )
1500    FORMAT(' INPUT DATE AND TIME')
1600    FORMAT(A9, A8)
1700    FORMAT (' THE FOLLOWING VARIABLES CAN BE PLOTTED VS ITERATION'/
     1       '  1.   ABSOLUTE ERROR '/
     2       '  2.   MAXIMUM ERROR  '/
     3       '  3.   RMS ERROR      ')
1800    FORMAT (5X,'XMIN = ',G14.5, 5X, 'XMAX = ',G14.5/
     1          5X,'YMIN = ',G14.5, 5X, 'YMAX = ',G14.5)
1900    FORMAT(A1)

        END
```

PLXSET

```
        SUBROUTINE PLXSET (IOPT$, INDGR)
C
        DIMENSION IOPT$ (*)
        SAVE IOPTP, INDGRP, XMIN, XMAX, YMIN, YMAX
C
C*****************************************************************
C
C       THIS SUBROUTINE SETS THE DISPLAY PARAMETERS FOR THE LINE CURVES
```

```
C       AND SETS THE SCALES OF THE PLOTS IN THE COMMON BLOCKS OF THE
C       GRAFIC ROUTINES.
C       *** CHECK ALL THE CALLING ROUTINES AND SET DEFAULT IOPT$(1)
C       BEFORE CALLING
C
C*********************************************************************
C
C       SET THE DEFAULT VALUES
C
        JTERMI = 5
        JTERMO = 6
C
        WRITE(JTERMO,20)
        READ (JTERMI,*) ITYPE

        IF (ITYPE .EQ. 0) THEN
          INDGR    = 21
          RETURN
        ENDIF

        IF (ITYPE .EQ. 4) THEN
          INDGR    = INDGRP
          IOPT$(1) = IOPTP
          IF (INDGR .EQ. 22) CALL GRSSET (XMIN, XMAX, YMIN, YMAX)
          RETURN
        ENDIF

        IF (ITYPE .EQ. 1) INDGR    = 21

        IF (ITYPE .EQ. 1 .OR. ITYPE .EQ. 3) THEN
          WRITE(JTERMO,30)
          READ (JTERMI,*) IOPT$(1)
        ENDIF

        IF (ITYPE .EQ. 2 .OR. ITYPE .EQ. 3) THEN
          INDGR    = 22
          WRITE(JTERMO,40)
          READ (JTERMI,* ) XMIN, XMAX, YMIN, YMAX
          CALL GRSSET (XMIN, XMAX, YMIN, YMAX)
          WRITE(JTERMO,50) XMIN, XMAX, YMIN, YMAX
        ENDIF

        INDGRP = INDGR
        IOPTP  = IOPT$(1)

        RETURN
C
C       -----------------
C       format statements
C       -----------------
C
20      FORMAT ( 5X, 'INPUT THE PLOT VARIABLES',/
     1          10X, '0.  USE DEFAULT VALUES'/
     2          10X, '1.  SET FORM OF THE DISPLAY OF THE CURVES'/
     3          10X, '2.  SET SCALES OF THE CURVES'/
     4          10X, '3.  BOTH 2. AND 3. '/
     5          10X, '4.  USE PREVIOUS VALUES '/
```

```
     6          10X, ' ==> ',$)
30      FORMAT ( 5X, 'INPUT THE PLOT DISPLAY PARAMETER'/
     1           5X, 'USE THE FOLLOWING OR THEIR COMBINATION'/
     2          10X, '1.  CLOSED CURVE'/
     3          10X, '2.  SOLID LINE'/
     4          10X, '4.  SYMBOLS'/
     5          10X, ' ==> ',$)
40      FORMAT ( 5X, 'INPUT THE SCALE VALUES XMIN, XMAX, YMIN, YMAX'/
     1          10X, ' ==> ',$)
50      FORMAT(10X,'XMIN = ', G14.5, 10X, 'XMAX = ',G14.5/
     1         10X,'YMIN = ', G14.5, 10X, 'YMAX = ',G14.5)

        END
```

ZRDUMY

```
        SUBROUTINE ZRDUMY (IFUN, INDGR, PLTITL, ALIMITS, ISTRING,
     1     A1,A2,A3,A4,A5,A6,A7,A8,A9,A10)
        CHARACTER PLTITL*(*), ISTRING*(*)
        DIMENSION ALIMITS(*)
        RETURN
        END
```

ZRPLTC

```
        SUBROUTINE ZRPLTC (IFUN, INDGR, PLTITL, ALIMITS, ISTRING,
     1    MBNDG2,ICELG2,ICELA2,ZX,ZY,ZF,GRDUMY,IBNDG2,KAUXG2,IMARKN)
C
        DIMENSION ZX(*), ZY(*), ZF(*), GRDUMY(*), ALIMITS(*),
     1            ICELA2(*), IMARKN(*), KAUXG2(*)

        DIMENSION ICELG2(10,*), IBNDG2(5,MBNDG2)
        REAL*4    ZX, ZY, ZF, GRDUMY, ALIMITS
        CHARACTER PLTITL*96 , ISTRING*80, NUMBER*10
        CHARACTER ctime*8
c       INTEGER*2 ICELG2, ICELA2, IBNDG2
        DIMENSION RX(4),RY(4),XCOR(2,4),FCOR(4),ITWO(8),
     1            FV(3),XV(3),YV(3),XP(9),YP(9),FP(9)
C       include '[pervaiz.twod0.inc]grcomn.inc'
        include '[pervaiz.grafic1]mpcomn.inc'
        common /ast$$$/ astc$$, asty$$
        logical astc$$, asty$$
        DATA ITWO /3,4,5,6,7,8,9,2/
C
C*********************************************************************
C
C       THIS SUBROUTINE GENERATES A GRID PLOT FOR THE GRID
C       CONTAINED IN /G2COMN/
C
C*********************************************************************
C
        GOTO (1000, 2000, 3000, 4000, 5000), IFUN+1
C
```

```
C       -------------
C       INITIALIZATION
C       -------------
C
1000    CONTINUE
        NCELA2 = NINT(GRDUMY(1))
        NNODG2 = NINT(GRDUMY(2))
        NCELG2 = NINT(GRDUMY(3))
        NBNDG2 = NINT(GRDUMY(4))
        XMIN   = GRDUMY( 5)
        XMAX   = GRDUMY( 6)
        YMIN   = GRDUMY( 7)
        YMAX   = GRDUMY( 8)
        ZMIN   = GRDUMY(13)
        ZMAX   = GRDUMY(14)
        ILABEL = NINT(GRDUMY(15))
        INDDFM = NINT(GRDUMY(16))
        IMIN   = NINT(GRDUMY(27))
        IMAX   = NINT(GRDUMY(28))

        rtime  = grdumy(26)
        ctime=' '
        if (rtime .ge. 0.) then
        if (10.*rtime .ge. 1.) then
          write(ctime,1010) rtime
        else
          write(ctime,1020) rtime
        endif
        endif
1010    format('t = ',f3.1)
1020    format('t = ',f4.2)

        JTERMO = 6
        JTERMI = 5

        RETURN
C       ----------------------------------------------------------------
C       ----------------------------------------------------------------
C
C       ----------------------
C       GET LIMITS OF THE DATA
C       ----------------------
C
2000    CONTINUE
C       CALL GR_GET_LIMITS (ZX,ZY,NNODG2,ALIMITS)
        ALIMITS(1) = XMIN
        ALIMITS(2) = XMAX
        ALIMITS(3) = YMIN
        ALIMITS(4) = YMAX
        ALIMITS(5) = ZMIN
        ALIMITS(6) = ZMAX
        ISTRING = ' MINIMUM AND MAXIMUM COUNTOUR VALUES:'
        RETURN
C       ----------------------------------------------------------------
C       ----------------------------------------------------------------
C
C       -------------------------
```

```
C       GET THE VALUE OF A CONTOUR
C       --------------------------
C
C       THE POSITION OF THE POINT IS STORED IN (ALIMITS(1),ALIMITS(2))
C
3000    CONTINUE

        VAL = 0
        DO 3010 JCELL = 1, NCELA2

C          FIND THE ACTUAL CELL NUMBER

           ICELL = ICELA2(JCELL)
C
C          STORE CORNERS OF BOX
C
           XCOR(1,1) = ZX(ICELG2(2,ICELL))
           XCOR(2,1) = ZY(ICELG2(2,ICELL))
           FCOR(1)   = ZF(ICELG2(2,ICELL))
C
           XCOR(1,2) = ZX(ICELG2(4,ICELL))
           XCOR(2,2) = ZY(ICELG2(4,ICELL))
           FCOR(2)   = ZF(ICELG2(4,ICELL))
C
           XCOR(1,3) = ZX(ICELG2(6,ICELL))
           XCOR(2,3) = ZY(ICELG2(6,ICELL))
           FCOR(3)   = ZF(ICELG2(6,ICELL))
C
           XCOR(1,4) = ZX(ICELG2(8,ICELL))
           XCOR(2,4) = ZY(ICELG2(8,ICELL))
           FCOR(4)   = ZF(ICELG2(8,ICELL))
C
C          SEE IF THE POINT IS IN THE CELL UNDER CONSIDERATION,
C          AND IF SO DETERMINE THE VALUE OF THE CONTOUR HERE
C
           CALL GR_INSIDE ( IIN, XCOR, 4, ALIMITS(1), ALIMITS(2) )
           IF (IIN .EQ. 1) THEN
              CALL GR_CONTOUR_VALUE (XCOR, FCOR,
     1           ALIMITS(1), ALIMITS(2), VAL)
              GOTO 3020
           ENDIF
3010    CONTINUE
3020    WRITE (ISTRING,3030) VAL
3030    FORMAT (' Function value =',G15.6)
        RETURN
C       ------------------------------------------------------------------
C       ------------------------------------------------------------------
C
C       -------------
C       PLOT CONTOURS
C       -------------
C
C       ONLY THE CONTOURS FOR THE CEWIC CELLS WILL BE PRODUCED

4000    CONTINUE
c       call set_astc
```

```
c       IMAX = 210
c       IMIN = 60
        DI     = IMAX - IMIN
        DF     = ZMIN - ZMAX
        DI_DF  = DI/DF
C
C       SETUP THE CLIPPING WINDOW AND VIEWPORT (SEE GKDISAT FOR DIMENSIONS)
C       AA=XGKOFF; CC=YGKOFF; BB=N-XGKOFF; DD=M-YGKOFF
C
        AA = 0.5
        BB = 8.5
        CC = 0.5
        DD = 6.5
        AA = XGKOFF
        BB = XGKMAX - XGKOFF
        CC = YGKOFF
        DD = YGKMAX - YGKOFF
        ZGKMAX = MAX (XGKMAX,YGKMAX)
C       LEAVE SPACE FOR COLOR KEY IF NEED BE
        IF (ILABEL .NE. 0) BB = BB - 1.

        CALL GKS$SET_WINDOW(2,AA,BB,CC,DD)
C       AA = AA/9.
C       BB = BB/9.
C       CC = CC/9.
C       DD = DD/9.

        AA = AA/ZGKMAX
        BB = BB/ZGKMAX
        CC = CC/ZGKMAX
        DD = DD/ZGKMAX
        CALL GKS$SET_VIEWPORT(2,AA,BB,CC,DD)
        CALL GKS$SELECT_XFORM(2)
        CALL GR_GET_SCALE (XMINGR,XMAXGR,YMINGR,YMAXGR)
C
C       STEP THROUGH EACH BOX
C
        DO 4090 JCELL = 1, NCELA2

                if (astc$$) then
                        write(6,*) jcell
                        astc$$ = .false.
                        call set_astc
                        goto 4092
                end if

C          FIND THE ACTUAL CELL NUMBER

           ICELL = ICELA2(JCELL)

C          SET THE POINTERS FOR THIS CELL

           KSW = ICELG2(2,ICELL)
           KS  = ICELG2(3,ICELL)
           KSE = ICELG2(4,ICELL)
           KE  = ICELG2(5,ICELL)
           KNE = ICELG2(6,ICELL)
```

```
            KN  = ICELG2(7,ICELL)
            KNW = ICELG2(8,ICELL)
            KW  = ICELG2(9,ICELL)
C
C           STORE CORNERS OF BOX
C
            XP(2) = ZX(KSW)
            YP(2) = ZY(KSW)
            FP(2) = ZF(KSW)
C
            XP(4) = ZX(KSE)
            YP(4) = ZY(KSE)
            FP(4) = ZF(KSE)
C
            XP(6) = ZX(KNE)
            YP(6) = ZY(KNE)
            FP(6) = ZF(KNE)
C
            XP(8) = ZX(KNW)
            YP(8) = ZY(KNW)
            FP(8) = ZF(KNW)
C
C           CHECK THE LIMITS OF THE RECTANGLE
C
            XMIN = MIN (XP(2),XP(4),XP(6),XP(8))
            XMAX = MAX (XP(2),XP(4),XP(6),XP(8))
            YMIN = MIN (YP(2),YP(4),YP(6),YP(8))
            YMAX = MAX (YP(2),YP(4),YP(6),YP(8))
            FMIN = MIN (FP(2),FP(4),FP(6),FP(8))
            FMAX = MAX (FP(2),FP(4),FP(6),FP(8))
C
            IF (XMAX.LT.XMINGR .OR. XMIN.GT.XMAXGR) GOTO 4090
            IF (YMAX.LT.YMINGR .OR. YMIN.GT.YMAXGR) GOTO 4090
C
C           CHECK IF THE WHOLE QUADRILATERAL CAN BE COLORED
C
            FRAT   = (FMIN-FMAX)*DI_DF
            INDDIF = FRAT
            FP(1)  = 0.25*(FP(2)+FP(4)+FP(6)+FP(8))

            IF (INDDIF .LT. 3) THEN
               FRAT   = (FP(1)-ZMAX)*DI_DF
               INDDIF = IMIN + FRAT
               XP(1)  = XP(2)
               XP(2)  = XP(4)
               XP(3)  = XP(6)
               XP(4)  = XP(8)
               YP(1)  = YP(2)
               YP(2)  = YP(4)
               YP(3)  = YP(6)
               YP(4)  = YP(8)
               CALL GR_FILL(INDDIF,1,XP,YP,4)
               GOTO 4090
            ENDIF
C
C           NOW STORE EDGES OF BOX
C
```

```
            IF (KS .NE. 0) THEN
              XP(3) = ZX(KS)
              YP(3) = ZY(KS)
              FP(3) = ZF(KS)
            ELSE
              XP(3) = 0.5*(XP(2)+XP(4))
              YP(3) = 0.5*(YP(2)+YP(4))
              FP(3) = 0.5*(FP(2)+FP(4))
            ENDIF
C
            IF (KE .NE. 0) THEN
              XP(5) = ZX(KE)
              YP(5) = ZY(KE)
              FP(5) = ZF(KE)
            ELSE
              XP(5) = 0.5*(XP(6)+XP(4))
              YP(5) = 0.5*(YP(6)+YP(4))
              FP(5) = 0.5*(FP(6)+FP(4))
            ENDIF
C
            IF (KN .NE. 0) THEN
              XP(7) = ZX(KN)
              YP(7) = ZY(KN)
              FP(7) = ZF(KN)
            ELSE
              XP(7) = 0.5*(XP(8)+XP(6))
              YP(7) = 0.5*(YP(8)+YP(6))
              FP(7) = 0.5*(FP(8)+FP(6))
            ENDIF
C
            IF (KW .NE. 0) THEN
              XP(9) = ZX(KW)
              YP(9) = ZY(KW)
              FP(9) = ZF(KW)
            ELSE
              XP(9) = 0.5*(XP(2)+XP(8))
              YP(9) = 0.5*(YP(2)+YP(8))
              FP(9) = 0.5*(FP(2)+FP(8))
            ENDIF
C
C           NOW STORE CENTER OF BOX
C
            XP(1) = 0.25*(XP(2)+XP(4)+XP(6)+XP(8))
            YP(1) = 0.25*(YP(2)+YP(4)+YP(6)+YP(8))
C
C           DIVIDE THE CELL INTO EIGHT TRIANGLES
C
            DO ITRI = 2, 9
               INXT = ITWO(ITRI-1)
               XV(1) = XP(ITRI)
               YV(1) = YP(ITRI)
               FV(1) = FP(ITRI)

               XV(2) = XP(INXT)
               YV(2) = YP(INXT)
               FV(2) = FP(INXT)
```

```
            XV(3) = XP(1)
            YV(3) = YP(1)
            FV(3) = FP(1)
            CALL GR_SMCOL_TRIN (XV,YV,FV,zmIN,zmAX,IMAX,IMIN,INDDFM)
          ENDDO
C
C         GO ONTO NEXT BOX
C
4090    CONTINUE
C
C       RESET THE CLIPPING VALUES
C
4092    CALL GKS$SELECT_XFORM(1)
C
C       CHANGE THE CLIPPING PARAMETER FOR THE RIGHT BOUNDARY
C
        IF(ILABEL.NE.0) THEN
            CALL GR_GET_CLIP  (XMNCGR,XMXCGR,YMNCGR,YMXCGR)
            XMXCOL = XMXCGR
            XMXCGR = XMAXGR - 0.14*(XMAXGR-XMINGR)
            CALL GR_SET_CLIP (XMNCGR,XMXCGR,YMNCGR,YMXCGR)
        ENDIF
C
C       DRAW THE BOUNDARIES
C
        DO 4095 IB = 1, NCELG2
          IMARKN(IB) = 0
4095    CONTINUE

        DO 4060 IB = 1, NBNDG2
           INODE = IBNDG2(1,IB)
           NCEL1 = IBNDG2(2,IB)
           NCEL2 = IBNDG2(3,IB)
           IEDGE = IBNDG2(4,IB)
           IF (NCEL1 .EQ. 0) GOTO 4060

C          CHECK THE CORNER CELLS

           IF (NCEL2 .EQ. 0) THEN
C             IF (IMARKN(NCEL1) .EQ. -1 .OR. KAUXG2(NCEL1).EQ.0)
C     1            GOTO 4060
              IF (KAUXG2(NCEL1).EQ.0) GOTO 4060
              IMARKN(NCEL1) = -1
              DO 4030 IED = 2, 8, 2
                 IF (IED .EQ. 8) THEN
                    INX = 2
                 ELSE
                    INX = IED + 2
                 ENDIF
                 IF (IEDGE .EQ. IED .OR. IEDGE .EQ. INX) THEN
                    XED = ZX(ICELG2(IED,NCEL1))
                    YED = ZY(ICELG2(IED,NCEL1))
                    XNX = ZX(ICELG2(INX,NCEL1))
                    YNX = ZY(ICELG2(INX,NCEL1))
                    CALL GK_MOVE (XED, YED, 0)
                    CALL GK_DRAW (XNX, YNX, 0)
                 ENDIF
```

```
4030        CONTINUE
            GOTO 4060
          ENDIF

C         CHECK THE EDGE CELLS
C
          IF (IMARKN(NCEL1) .EQ. -1 .OR. KAUXG2(NCEL1).EQ.0)
     1          GOTO 4050
          IMARKN(NCEL1) = -1
          DO 4050 IED = 3, 9, 2
             IBG = IED - 1
             IF (IBG .EQ. 8) THEN
                INX = 2
             ELSE
                INX = IBG + 2
             ENDIF
             IF (IEDGE .EQ. IED) THEN
                XBG = ZX(ICELG2(IBG,NCEL1))
                YBG = ZY(ICELG2(IBG,NCEL1))
                XNX = ZX(ICELG2(INX,NCEL1))
                YNX = ZY(ICELG2(INX,NCEL1))
                CALL GK_MOVE (XBG, YBG, 0)
                CALL GK_DRAW (XNX, YNX, 0)
             ENDIF
4050      CONTINUE

          IF (IMARKN(NCEL2) .EQ. -1 .OR. KAUXG2(NCEL2).EQ.0)
     1          GOTO 4060
          IMARKN(NCEL2) = -1
          DO 4055 IED = 3, 9, 2
             IBG = IED - 1
             IF (IBG .EQ. 8) THEN
                INX = 2
             ELSE
                INX = IBG + 2
             ENDIF
             IF (IEDGE .EQ. IED) THEN
                XBG = ZX(ICELG2(IBG,NCEL2))
                YBG = ZY(ICELG2(IBG,NCEL2))
                XNX = ZX(ICELG2(INX,NCEL2))
                YNX = ZY(ICELG2(INX,NCEL2))
                CALL GK_MOVE (XBG, YBG, 0)
                CALL GK_DRAW (XNX, YNX, 0)
             ENDIF
4055      CONTINUE

4060    CONTINUE
C
C       IF LABELING IS REQUIRED, DRAW THE KEY
C
        IF(ILABEL.NE.0) THEN
            CALL GR_SET_CLIP (XMNCGR,XMXCOL,YMNCGR,YMXCGR)
            XB1 = XMAXGR - 0.120*(XMAXGR-XMINGR)
            XB2 = XMAXGR - 0.080*(XMAXGR-XMINGR)
            YT  = YMAXGR - 2./31.0*(YMAXGR-YMINGR)
            YB  = YMINGR + 2./31.0*(YMAXGR-YMINGR)
            DI  = IMAX - IMIN
```

```
            DY  = (YT-YB)/DI
            YTT = YT
C
            DO 5030 INDEX = IMIN, IMAX
              RX(1) = XB1
              RX(2) = XB2
              RX(3) = XB2
              RX(4) = XB1
              RY(1) = YT
              RY(2) = YT
              YT    = YT-DY
              RY(3) = YT
              RY(4) = YT
              CALL GR_FILL(INDEX,1,RX,RY,4)
5030        CONTINUE

            CALL GK_MOVE(XB1,YT,0)
            CALL GK_DRAW(XB2,YT,0)
            CALL GK_DRAW(XB2,YTT,0)
            CALL GK_DRAW(XB1,YTT,0)
            CALL GK_DRAW(XB1,YT,0)

            YT    = YTT
C           XB2   = XMAXGR - 0.060*(XMAXGR-XMINGR)
            NMARK =10
            DY1   = (YT-YB)/NMARK
            DO 5040 IND = 1, NMARK+1
                YRAT = (YT-YTT)/(YB-YTT)
                FNO  = zmAX + YRAT*(zmIN-zmAX)
                IF (ABS(FNO) .LT. 1.E7) THEN
                   WRITE(NUMBER,5050) FNO
                ELSE
                   WRITE(NUMBER,5070) FNO
                ENDIF
                CALL GK_MOVE(XB2,YT-DY,0)
                CALL GR_ANNOTATE(NUMBER)
                YT = YT - DY1
5040         CONTINUE
            XMXCGR = XMAXGR - 0.14*(XMAXGR-XMINGR)
            CALL GR_SET_CLIP (XMNCGR,XMXCGR,YMNCGR,YMXCGR)
          ENDIF

5050      FORMAT(F8.2)
5070      FORMAT(G10.2)

          if (rtime .ge. 0) then
          write(6,*) ' xb1 xmaxgr xmingr',xb1,xmaxgr,xmingr
            xb1 = xb1 - 0.25*(xmaxgr-xmingr)
            ytt = ytt - 2./31.0*(YMAXGR-YMINGR)
                  CALL GK_MOVE(XB1,YTT,0)
                  CALL GR_ANNOTATE(ctime)
          endif

5000      RETURN
          END
```

ZRPLTG

```
      SUBROUTINE ZRPLTG (IFUN, INDGR, PLTITL, ALIMITS, ISTRING,
     1    ICELG2,ICELA2,KAUXG2,ZX,ZY,GRDUMY,Z7,Z8,Z9,Z10)
C
      INCLUDE '[PERVAIZ.TWODO.INC] HEXCOD.INC     '
      DIMENSION ZX(*), ZY(*), GRDUMY(*), ALIMITS(*), ICELA2(*)
      DIMENSION ICELG2(10,*), KAUXG2(*)
      REAL*4    ZX, ZY, GRDUMY, ALIMITS
      DIMENSION XCOR(2,4)
      common /ast$$$/ astc$$, asty$$
      logical astc$$, asty$$
      CHARACTER PLTITL*96 , ISTRING*80
c     INTEGER*2 ICELG2, ICELA2
C
C*********************************************************************
C
C     THIS SUBROUTINE GENERATES A GRID PLOT FOR THE GRID
C     CONTAINED IN /G2COMN/
C
C*********************************************************************
C
      JTERMI  = 5
      JTERMO  = 6
      NCELA2  = NINT(GRDUMY(1))
      NNODG2  = NINT(GRDUMY(2))
      XMIN    = GRDUMY( 5)
      XMAX    = GRDUMY( 6)
      YMIN    = GRDUMY( 7)
      YMAX    = GRDUMY( 8)
      OFFSETX = GRDUMY(24)
      OFFSETY = GRDUMY(25)

      GOTO (1000,2000,3000,4000,5000),IFUN+1
C
C     --------------
C     INITIALIZATION
C     --------------
C
1000  RETURN
C
C     ----------------------
C     GET LIMITS OF THE DATA
C     ----------------------
C
2000  CONTINUE
C
      ALIMITS(1) = XMIN
      ALIMITS(2) = XMAX
      ALIMITS(3) = YMIN
      ALIMITS(4) = YMAX
      ALIMITS(5) = NCELA2
      ALIMITS(6) = GRDUMY(4)
```

```
      ISTRING    = ' TOTAL NUMBER OF CEWIC CELLS AND BOUNDARY NODES'
      RETURN
C
C     ------------------------
C     GET THE VALUE OF THE CELL
C     ------------------------
C
C     THE POSITION OF THE POINT IS STORED IN (ALIMITS(1),ALIMITS(2))
C
3000  CONTINUE
      ALIMITS(1) = ALIMITS(1) - OFFSETX
      ALIMITS(2) = ALIMITS(2) - OFFSETY
      ICVAL  = 0
      ICORN1 = 0
      ICORN2 = 0
      ICORN3 = 0
      ICORN4 = 0

      DO 3010 JCELL = 1, NCELA2

            if (astc$$) then
                  write(6,*) jcell
                  astc$$ = .false.
                  call set_astc
                  goto 3020
            end if

C        FIND THE ACTUAL CELL NUMBER

         ICELL = ICELA2(JCELL)
C
C        STORE CORNERS OF BOX
C
         XCOR(1,1) = ZX(ICELG2(2,ICELL))
         XCOR(2,1) = ZY(ICELG2(2,ICELL))
C
         XCOR(1,2) = ZX(ICELG2(4,ICELL))
         XCOR(2,2) = ZY(ICELG2(4,ICELL))
C
         XCOR(1,3) = ZX(ICELG2(6,ICELL))
         XCOR(2,3) = ZY(ICELG2(6,ICELL))
C
         XCOR(1,4) = ZX(ICELG2(8,ICELL))
         XCOR(2,4) = ZY(ICELG2(8,ICELL))
C
C        SEE IF THE POINT IS IN THE CELL UNDER CONSIDERATION,
C        AND IF SO DETERMINE THE VALUE OF THE CELL HERE
C
         CALL GR_INSIDE ( IIN, XCOR, 4, ALIMITS(1), ALIMITS(2) )
         IF (IIN .EQ. 1) THEN
            ICVAL  = ICELL
            ICORN1 = ICELG2(2,ICELL)
            ICORN2 = ICELG2(4,ICELL)
            ICORN3 = ICELG2(6,ICELL)
            ICORN4 = ICELG2(8,ICELL)
            GOTO 3020
         ENDIF
```

```
3010    CONTINUE

C       SEE IF DETAILED INFORMATION ABOUT THE CELL IS NEEDED FOR THE
C       INTERACTIVE GRID GENERATOR PROGRAM

3020    IF (ALIMITS(6) .LT. 0) THEN
           ISTRING    = ' '
           ALIMITS(5) = ICVAL
           RETURN
        ENDIF
C
C       USUAL INFORMATION
        WRITE (ISTRING,3030) ICVAL,ICORN1,ICORN2,ICORN3,ICORN4
3030    FORMAT (' CELL VALUE =',I6,5X,'CELL CORNER POINTERS :',4I7)
        RETURN
C
C       ----------
C       PLOT GRIDS
C       ----------
C
4000    CONTINUE

C       LOOP THROUGH ALL FOUR CORNERS OF THE CELLS AT THIS LEVEL
C       AND DRAW THE GRIDS

        DO 4010 JCELL = 1, NCELA2

              if (astc$$) then
                     write(6,*) jcell
                     astc$$ = .false.
                     call set_astc
                     goto 4020
              end if

C          FIND THE ACTUAL CELL NUMBER

           ICELL = ICELA2(JCELL)

C          MOVE TO THE NORTHWEST CORNER OF THIS CELL

           INODE  = ICELG2(8,ICELL)
           ZXNODE = ZX (   INODE) + OFFSETX
           ZYNODE = ZY (   INODE) + OFFSETY
           KX     = KAUXG2(ICELL)
           CALL   GK_MOVE(ZXNODE,ZYNODE,0)

C          DRAW TO ALL FOUR CORNERS OF THIS CELL
C
           INODE  = ICELG2(2,ICELL)
           ZXNODE = ZX (INODE) + OFFSETX
           ZYNODE = ZY (INODE) + OFFSETY
           CALL   GK_DRAW(ZXNODE,ZYNODE,0)

           INODE  = ICELG2(4,ICELL)
           ZXNODE = ZX (INODE) + OFFSETX
           ZYNODE = ZY (INODE) + OFFSETY
           CALL   GK_DRAW(ZXNODE,ZYNODE,0)
```

```
            IF(IAND(KX,KLOOOF).NE.0) THEN
              INODE  = ICELG2(6,ICELL)
              ZXNODE = ZX (INODE) + OFFSETX
              ZYNODE = ZY (INODE) + OFFSETY
              CALL   GK_DRAW(ZXNODE,ZYNODE,0)
            ENDIF

            IF(IAND(KX,KLOOOF).NE.0) THEN
              INODE  = ICELG2(8,ICELL)
              ZXNODE = ZX (INODE) + OFFSETX
              ZYNODE = ZY (INODE) + OFFSETY
              CALL   GK_DRAW(ZXNODE,ZYNODE,0)
            ENDIF

4010    CONTINUE
4020    RETURN
C
5000    CONTINUE
        CALL GR_REAL('INPUT X-OFFSET',OFFSETX)
        CALL GR_REAL('INPUT Y-OFFSET',OFFSETY)
        GRDUMY(24) = OFFSETX
        GRDUMY(25) = OFFSETY
C
        RETURN
        END
```

ZRPLTL

```
        SUBROUTINE ZRPLTL (IFUN, INDGR, PLTITL, ALIMITS, ISTRING,
     1     ICELG2,ICELA2,ZX,ZY,ZF,GRDUMY,CONT,IBNDG2,KAUXG2,IMARKN)
C
        DIMENSION ZX(*), ZY(*), ZF(*), GRDUMY(*), ALIMITS(*),
     1            ICELA2(*), CONT(*) , IMARKN(*), KAUXG2(*)
        DIMENSION XCOR(2,4), FCOR(4), CCHNGE(60), INDCHN(60)
        DIMENSION ICELG2(10,*), IBNDG2(5,*)
        REAL*4    ZX, ZY, ZF, GRDUMY, ALIMITS, CONT
        common /ast$$$/ astc$$, asty$$
        logical astc$$, asty$$
        CHARACTER PLTITL*96 , ISTRING*80, NUMBER*10
c       INTEGER*2 ICELG2, ICELA2, IBNDG2
C
C*********************************************************************
C
C       THIS SUBROUTINE GENERATES A GRID PLOT FOR THE GRID
C       CONTAINED IN /G2COMN/
C
C*********************************************************************
C
        JTERMO = 6
        JTERMI = 5
        NCELA2 = NINT(GRDUMY( 1))
        NNODG2 = NINT(GRDUMY( 2))
```

```
      NCELG2 = NINT(GRDUMY( 3))
      NBNDG2 = NINT(GRDUMY( 4))
      NCONTS = NINT(GRDUMY(17))
      NLABEL = NINT(GRDUMY(20))
      ICINC  = NINT(GRDUMY(21))
      INTERF = NINT(GRDUMY(22))
      XMIN   = GRDUMY( 5)
      XMAX   = GRDUMY( 6)
      YMIN   = GRDUMY( 7)
      YMAX   = GRDUMY( 8)
      ZMIN   = GRDUMY(13)
      ZMAX   = GRDUMY(14)
      ZCBASE = GRDUMY(18)
      ZCSTEP = GRDUMY(19)
      OFFSETX = GRDUMY(24)
      OFFSETY = GRDUMY(25)

C
      GOTO (1000,2000,3000,4000,5000),IFUN+1
C
C     --------------
C     INITIALIZATION
C     --------------
C
1000  WRITE(JTERMO,1010) NCONTS
1010  FORMAT(5X,'THE CONTOUR LEVELS ARE :',I5)

      DO 1030 I = 1, NCONTS
        CONT(I) = ZCBASE + (I-1)*ZCSTEP
1030  CONTINUE
      WRITE(JTERMO,1020) ((I, CONT(I)),I=1,NCONTS)
1020  FORMAT(3(1X,I2,2X,'CONT=',G13.4))

      NCHNGE = 0
1040  WRITE(JTERMO,1050)
1050  FORMAT(5X,'INPUT NUMBER OF CONTOUR VALUE TO BE CHANGED OR 0')
      READ (JTERMI,*) ICHNGE
      IF (ICHNGE .EQ. 0) THEN
        RETURN
      ELSE
        NCHNGE         = NCHNGE + 1
        INDCHN(NCHNGE) = ICHNGE
        READ (JTERMI,*) CCHNGE(NCHNGE)
        GOTO 1040
      ENDIF
C
C     ----------------------
C     GET LIMITS OF THE DATA
C     ----------------------
C
2000  CONTINUE
C     CALL GR_GET_LIMITS (ZX,ZY,NNODG2,ALIMITS)
      ALIMITS(1) = XMIN
      ALIMITS(2) = XMAX
      ALIMITS(3) = YMIN
      ALIMITS(4) = YMAX
      ALIMITS(5) = ZMIN
```

```
      ALIMITS(6) = ZMAX
      ISTRING = ' MINIMUM AND MAXIMUM COUNTOUR VALUES:'
      RETURN
C
C     -------------------------
C     GET THE VALUE OF A CONTOUR
C     -------------------------
C
C     THE POSITION OF THE POINT IS STORED IN (ALIMITS(1),ALIMITS(2))
C
3000  CONTINUE

      VAL = 0
      DO 3010 JCELL = 1, NCELA2

         if (astc$$) then
             write(6,*) jcell
             astc$$ = .false.
             call set_astc
             goto 3020
         end if

C        FIND THE ACTUAL CELL NUMBER

         ICELL = ICELA2(JCELL)
C
C        STORE CORNERS OF BOX
C
         XCOR(1,1) = ZX(ICELG2(2,ICELL))
         XCOR(2,1) = ZY(ICELG2(2,ICELL))
         FCOR(1)   = ZF(ICELG2(2,ICELL))
C
         XCOR(1,2) = ZX(ICELG2(4,ICELL))
         XCOR(2,2) = ZY(ICELG2(4,ICELL))
         FCOR(2)   = ZF(ICELG2(4,ICELL))
C
         XCOR(1,3) = ZX(ICELG2(6,ICELL))
         XCOR(2,3) = ZY(ICELG2(6,ICELL))
         FCOR(3)   = ZF(ICELG2(6,ICELL))
C
         XCOR(1,4) = ZX(ICELG2(8,ICELL))
         XCOR(2,4) = ZY(ICELG2(8,ICELL))
         FCOR(4)   = ZF(ICELG2(8,ICELL))
C
C        SEE IF THE POINT IS IN THE CELL UNDER CONSIDERATION,
C        AND IF SO DETERMINE THE VALUE OF THE CONTOUR HERE
C
         CALL GR_INSIDE ( IIN, XCOR, 4, ALIMITS(1), ALIMITS(2) )
         IF (IIN .EQ. 1) THEN
            CALL GR_CONTOUR_VALUE (XCOR, FCOR,
     1                 ALIMITS(1), ALIMITS(2), VAL)
            GOTO 3020
         ENDIF
3010  CONTINUE
3020  WRITE (ISTRING,3030) VAL
3030  FORMAT (' Function value =',G15.6)
      RETURN
```

```
C          -
C       ----------
C       PLOT CONTOURS
C       ------------
C
C       ONLY THE CONTOURS FOR THE CEWIC CELLS WILL BE PRODUCED

4000    CONTINUE
        if (xoffset .ne. 0. .or. yoffset .ne. 0.) goto 4001
C
C       IF LABELING IS REQUIRED, DRAW THE KEY
C
        IF (NLABEL .NE. 0) THEN
           CALL GR_GET_SCALE (XMINGR, XMAXGR, YMINGR, YMAXGR)
           CALL GR_GET_CLIP  (XMNCGR, XMXCGR, YMNCGR, YMXCGR)
           DO 4010 ICONT = 1, NLABEL
              XX = XMAXGR - 0.120*(XMAXGR-XMINGR)
              YY = YMAXGR - REAL(ICONT*3-1)/31.0*(YMAXGR-YMINGR)
              CALL GK_MOVE(XX, YY, ICONT)
              YY = YMAXGR - REAL(ICONT*3)/31.0*(YMAXGR-YMINGR)
              WRITE (NUMBER,6000) CONT(1+ICINC*(ICONT-1))
              CALL GK_MOVE     (XX, YY, 0)
              CALL GR_ANNOTATE ( NUMBER  )
4010       CONTINUE
C          CHANGE THE CLIPPING PARAMETER FOR THE RIGHT BOUNDARY
           XMXCGR = 0.125*XMINGR + 0.875*XMAXGR
           CALL GR_SET_CLIP (XMNCGR,XMXCGR,YMNCGR,YMXCGR)
        ENDIF
C
C       DRAW THE BOUNDARY, ADDING CONTOUR MARKINGS IF REQUIRED
C
4001    DO 4015 IB = 1, NCELG2
          IMARKN(IB) = 0
4015    CONTINUE

        DO 4060 IB = 1, NBNDG2

           if (astc$$) then
               write(6,*) jcell
               astc$$ = .false.
               call set_astc
               goto 4095
           end if

           INODE = IBNDG2(1,IB)
           NCEL1 = IBNDG2(2,IB)
           NCEL2 = IBNDG2(3,IB)
           IEDGE = IBNDG2(4,IB)
           IF (NCEL1 .EQ. 0) GOTO 4060
C          CHECK THE CORNER CELLS

           IF (NCEL2 .EQ. 0) THEN
C             IF (IMARKN(NCEL1) .EQ. -1 .OR. KAUXG2(NCEL1).EQ.0)
C     1                    GOTO 4060
              IF (KAUXG2(NCEL1).EQ.0) GOTO 4060
              IMARKN(NCEL1) = -1
              DO 4030 IED = 2, 8, 2
```

```
      -     IF (IED .EQ. 8) THEN
      -        INX = 2
            ELSE
               INX = IED + 2
            ENDIF
            IF (IEDGE .EQ. IED .OR. IEDGE .EQ. INX) THEN
               XED = ZX(ICELG2(IED,NCEL1)) + xoffset
               YED = ZY(ICELG2(IED,NCEL1)) + yoffset
               FED = ZF(ICELG2(IED,NCEL1))
               XNX = ZX(ICELG2(INX,NCEL1)) + xoffset
               YNX = ZY(ICELG2(INX,NCEL1)) + yoffset
               FNX = ZF(ICELG2(INX,NCEL1))
               CALL GK_MOVE (XED, YED, 0)
               CALL GK_DRAW (XNX, YNX, 0)
               DO 4020 ICONT = 1, NLABEL
                 FCONT = CONT((ICONT-1)*ICINC + 1)
                 CALL GR_CROSS (FED,FNX,FCONT,ALFA)
                 IF (ALFA.GE.0.0 .AND. ALFA.LE.1.0) THEN
                   XX = XED*(1.0-ALFA)+ZX(INX)*ALFA
                   YY = YED*(1.0-ALFA)+ZY(INX)*ALFA
                   CALL GK_MOVE(XX, YY, -ICONT)
                 ENDIF
4020           CONTINUE
               ENDIF
4030        CONTINUE
            GOTO 4060
          ENDIF

C         CHECK THE EDGE CELLS

          IF (IMARKN(NCEL1) .EQ. -1 .OR. KAUXG2(NCEL1).EQ.0)
     1                GOTO 4050
          IMARKN(NCEL1) = -1
          DO 4050 IED = 3, 9, 2
            IBG = IED - 1
            IF (IBG .EQ. 8) THEN
               INX = 2
            ELSE
               INX = IBG + 2
            ENDIF
            IF (IEDGE .EQ. IED) THEN
               XBG = ZX(ICELG2(IBG,NCEL1)) + xoffset
               YBG = ZY(ICELG2(IBG,NCEL1)) + yoffset
               FBG = ZF(ICELG2(IBG,NCEL1))
               XNX = ZX(ICELG2(INX,NCEL1)) + xoffset
               YNX = ZY(ICELG2(INX,NCEL1)) + yoffset
               FNX = ZF(ICELG2(INX,NCEL1))
               CALL GK_MOVE (XBG, YBG, 0)
               CALL GK_DRAW (XNX, YNX, 0)
               DO 4040 ICONT = 1, NLABEL
                  FCONT = CONT((ICONT-1)*ICINC + 1)
                  CALL GR_CROSS (FBG,FNX,FCONT,ALFA)
                  IF(ALFA.GE.0.0 .AND. ALFA.LE.1.0) THEN
                     XX = XBG*(1.0-ALFA)+XNX*ALFA
                     YY = YBG*(1.0-ALFA)+YNX*ALFA
                     CALL GK_MOVE(XX, YY, -ICONT)
                  ENDIF
```

```
4040        -    CONTINUE
             - ENDIF
4050       CONTINUE

           IF (IMARKN(NCEL2) .EQ. -1 .OR. KAUXG2(NCEL2).EQ.0)
     1                    GOTO 4060
           IMARKN(NCEL2) = -1
           DO 4055 IED = 3, 9, 2
              IBG = IED - 1
              IF (IBG .EQ. 8) THEN
                 INX = 2
              ELSE
                 INX = IBG + 2
              ENDIF
              IF (IEDGE .EQ. IED) THEN
                 XBG = ZX(ICELG2(IBG,NCEL2)) + xoffset
                 YBG = ZY(ICELG2(IBG,NCEL2)) + yoffset
                 FBG = ZF(ICELG2(IBG,NCEL2))
                 XNX = ZX(ICELG2(INX,NCEL2)) + xoffset
                 YNX = ZY(ICELG2(INX,NCEL2)) + yoffset
                 FNX = ZF(ICELG2(INX,NCEL2))
                 CALL GK_MOVE (XBG, YBG, 0)
                 CALL GK_DRAW (XNX, YNX, 0)
                 DO 4054 ICONT = 1, NLABEL
                    FCONT = CONT((ICONT-1)*ICINC + 1)
                    CALL GR_CROSS (FBG,FNX,FCONT,ALFA)
                    IF(ALFA.GE.0.0 .AND. ALFA.LE.1.0) THEN
                       XX = XBG*(1.0-ALFA)+XNX*ALFA
                       YY = YBG*(1.0-ALFA)+YNX*ALFA
                       CALL GK_MOVE(XX, YY, -ICONT)
                    ENDIF
4054             CONTINUE
              ENDIF
4055       CONTINUE

4060    CONTINUE
C
C       SLIGHTLY MODIFY THE CONTOUR ARRAY TO REMOVE NOISE
C
        IF (NCHNGE .NE. 0) THEN
          DO 4070 ICHNGE = 1, NCHNGE
             CONT(INDCHN(ICHNGE)) = CCHNGE(ICHNGE)
4070      CONTINUE
        ENDIF
C
C       STEP THROUGH EACH BOX
C
        DO 4090 JCELL = 1, NCELA2

           if (astc$$) then
                write(6,*) jcell
                astc$$ = .false.
                call set_astc
                goto 4095
           end if

C          FIND THE ACTUAL CELL NUMBER
```

```
            ICELL = ICELA2(JCELL)

C           SET THE POINTERS FOR THIS CELL

            KSW = ICELG2(2,ICELL)
            KS  = ICELG2(3,ICELL)
            KSE = ICELG2(4,ICELL)
            KE  = ICELG2(5,ICELL)
            KNE = ICELG2(6,ICELL)
            KN  = ICELG2(7,ICELL)
            KNW = ICELG2(8,ICELL)
            KW  = ICELG2(9,ICELL)
C
C           STORE CORNERS OF BOX
C
            XSW = ZX(KSW) + xoffset
            YSW = ZY(KSW) + yoffset
            FSW = ZF(KSW)
C
            XSE = ZX(KSE) + xoffset
            YSE = ZY(KSE) + yoffset
            FSE = ZF(KSE)
C
            XNE = ZX(KNE) + xoffset
            YNE = ZY(KNE) + yoffset
            FNE = ZF(KNE)
C
            XNW = ZX(KNW) + xoffset
            YNW = ZY(KNW) + yoffset
            FNW = ZF(KNW)
C
C           NOW STORE EDGES OF BOX
C
            IF (KS .NE. 0) THEN
              XS = ZX(KS)
              YS = ZY(KS)
              FS = ZF(KS)
            ELSE
              XS = 0.5*(XSW+XSE)
              YS = 0.5*(YSW+YSE)
              FS = 0.5*(FSW+FSE)
            ENDIF
C
            IF (KE .NE. 0) THEN
              XE = ZX(KE)
              YE = ZY(KE)
              FE = ZF(KE)
            ELSE
              XE = 0.5*(XNE+XSE)
              YE = 0.5*(YNE+YSE)
              FE = 0.5*(FNE+FSE)
            ENDIF
C
            IF (KN .NE. 0) THEN
              XN = ZX(KN)
              YN = ZY(KN)
```

```
            FN = ZF(KN)
          ELSE
            XN = 0.5*(XNW+XNE)
            YN = 0.5*(YNW+YNE)
            FN = 0.5*(FNW+FNE)
          ENDIF
C
          IF (KW .NE. 0) THEN
            XW = ZX(KW)
            YW = ZY(KW)
            FW = ZF(KW)
          ELSE
            XW = 0.5*(XSW+XNW)
            YW = 0.5*(YSW+YNW)
            FW = 0.5*(FSW+FNW)
          ENDIF
C
C         NOW STORE CENTER OF BOX
C
          XC = 0.25*(XSW+XSE+XNE+XNW)
          YC = 0.25*(YSW+YSE+YNE+YNW)
          FC = 0.25*(FSW+FSE+FNE+FNW)
C
C         STEP THROUGH EACH CONTOUR
C
          DO 4080 ICNT = 1, NCONTS
            CNT = CONT(ICNT)
            CALL GR_CBOX(XSW, YSW, FSW,  XS , YS , FS ,
     1                          XC , YC , FC ,  XW , YW , FW, CNT)
            CALL GR_CBOX(XS , YS , FS ,  XSE, YSE, FSE,
     1                          XE , YE , FE ,  XC , YC , FC , CNT)
            CALL GR_CBOX(XC , YC , FC ,  XE , YE , FE ,
     1                          XNE, YNE, FNE,  XN , YN , FN , CNT)
            CALL GR_CBOX(XW , YW , FW ,  XC , YC , FC ,
     1                          XN , YN , FN ,  XNW, YNW, FNW, CNT)
4080      CONTINUE
C
C         SEE IF YOU WANT TO DRAW INTERFACE MARKS
C
          IF (INTERF .NE. 0) THEN
C
             IF (KS .NE. 0) THEN
                CALL GK_MOVE (XS ,YS, 0)
                CALL GK_DRAW (XS ,YS, INTERF)
             ENDIF
C
             IF (KE .NE. 0) THEN
                CALL GK_MOVE (XE ,YE, 0)
                CALL GK_DRAW (XE ,YE, INTERF)
             ENDIF
C
             IF (KN .NE. 0) THEN
                CALL GK_MOVE (XN ,YN, 0)
                CALL GK_DRAW (XN ,YN, INTERF)
             ENDIF
C
             IF (KW .NE. 0) THEN
```

```
            CALL GK_MOVE (XW ,YW, 0)
            CALL GK_DRAW (XW ,YW, INTERF)
          ENDIF
C
        ENDIF
C
C       GO ONTO NEXT BOX
C
4090  CONTINUE
4095  CONTINUE

6000  FORMAT(F10.4)
      RETURN
5000  CONTINUE
C
      WRITE (JTERMO,1778)
1778  FORMAT(' INPUT THE NUMBER (NCONT) OF CONTOURS DESIRED:'/5X,
     1  '1. NCONT < 0           : ABS(NCONT) CONTOURS ARE PLOTTED'/5X,
     3  '2. 1000 < NCONT < 2000 : NCONT-2000 CONTOURS ARE PLOTTED'/
     4  5X, 'AUTOMATIC SCALING IS DONE FOR CASES 2 AND 3'            )
      READ (JTERMI,*) NCONT
C
      IF (NCONT .LT. 0) THEN
        WRITE (JTERMO,1800)
        READ  (JTERMI,   *) ZCBASE, ZCSTEP
      ENDIF
1800  FORMAT('  CONTOURS ARE DEFINED BY:'/5X,
     1   'CONTOUR(I)=ZCBASE + (I-1)*ZCSTEP ; I = 1 TO # CONTOURS'/
     2   5X,'INPUT ZCBASE AND ZCSTEP'                              )
C
C     NCONTS IS THE ACTUAL NUMBER OF CONTOURS,
C
      NC1    = ABS (NCONT)
      NCONTS = MOD (NC1,1000)
      NC2    = NC1 / 1000
C
      NLABEL = 0
      ICINC  = 0
C
C     FIND THE CONTOUR LEVELS (ZCBASE : BASE CONTOUR LEVEL,
C                              ZCSTEP : CONTOUR INCREMENT )
C
      IF (NCONT .GT. 0) THEN
        CALL GR_SCALE (ZMIN, ZMAX, NCONTS-1, ZCBASE, ZCSTEP)
      ENDIF
C
      GRDUMY(17) = NCONTS
      GRDUMY(18) = ZCBASE
      GRDUMY(19) = ZCSTEP
      GRDUMY(20) = NLABEL
      GRDUMY(21) = ICINC
      GRDUMY(22) = 0.

      CALL GR_REAL('INPUT X-OFFSET',OFFSETX)
      CALL GR_REAL('INPUT Y-OFFSET',OFFSETY)
      GRDUMY(24) = OFFSETX
      GRDUMY(25) = OFFSETY
```

```
      RETURN
      END
```

ZRVECT

```
      SUBROUTINE ZRVECT(IFUN,INDGR,PLTITL,ALIMITS,ISTRING,
     1     ZX,ZY,ZP,ZT,KAUXG2,GRDUMY,ICELG2,ICELA2,IPOINT,IMARKN)
C
c     IMPLICIT INTEGER (H)
      DIMENSION ZX(*),ZY(*),ZP(*),ZT(*),GRDUMY(*), XCOR(2,4),
     1          ALIMITS(4), FCOR(4), KAUXG2(*)
      CHARACTER PLTITL*96 , ISTRING*80
c     INTEGER*2 ICELG2(10,*),ICELA2(*),IPOINT(*),IMARKN(*)
      INTEGER ICELG2(10,*),ICELA2(*),IPOINT(*),IMARKN(*)
      INCLUDE '[PERVAIZ.TWODO.INC]HEXCOD.INC'

C
C*********************************************************************
C
C     THIS SUBROUTINE GENERATES VECTOR PLOTS
C
C*********************************************************************
C
      JTERMO = 6
      UMIN   = GRDUMY( 9)
      UMAX   = GRDUMY(10)
      VMIN   = GRDUMY(11)
      VMAX   = GRDUMY(12)
      NCELA2 = NINT(GRDUMY( 1))
      NNODG2 = NINT(GRDUMY( 2))
      NCELG2 = NINT(GRDUMY( 3))
      MAXNOD = NINT(GRDUMY(23))

      GOTO (1000,2000,3000,4000,5000) IFUN + 1
C
C     --------------
C     INITIALIZATION
C     --------------
C
1000  RETURN
C
C     ----------------------
C     GET LIMITS OF THE DATA
C     ----------------------
C
2000  CONTINUE
C     CALL GR_GET_LIMITS (ZX,ZY,MAXNOD ,ALIMITS)
      ALIMITS(1) = GRDUMY( 5)
      ALIMITS(2) = GRDUMY( 6)
      ALIMITS(3) = GRDUMY( 7)
      ALIMITS(4) = GRDUMY( 8)
      ALIMITS(5) = GRDUMY(11)
      ALIMITS(6) = GRDUMY(12)
```

```
      WRITE(ISTRING,2010) UMIN, UMAX
2010  FORMAT(10X,'UMIN=',G15.7,10X,'UMAX=',G15.7 )
      RETURN
C
C     --------------------
C     GET THE VECTOR VALUE
C     --------------------
C
C     THE POSITION OF THE POINT IS STORED IN (ALIMITS(1),ALIMITS(2))
C
3000  CONTINUE
C
      UVAL = 0
      VVAL = 0
C
      DO 3010 JCELL = 1, NCELA2

C        FIND THE ACTUAL CELL NUMBER

         ICELL = ICELA2(JCELL)
C
C        STORE CORNERS OF BOX
C
         XCOR(1,1) = ZX(ICELG2(2,ICELL))
         XCOR(2,1) = ZY(ICELG2(2,ICELL))
         XCOR(1,2) = ZX(ICELG2(4,ICELL))
         XCOR(2,2) = ZY(ICELG2(4,ICELL))
         XCOR(1,3) = ZX(ICELG2(6,ICELL))
         XCOR(2,3) = ZY(ICELG2(6,ICELL))
         XCOR(1,4) = ZX(ICELG2(8,ICELL))
         XCOR(2,4) = ZY(ICELG2(8,ICELL))
C
C        SEE IF THE POINT IS IN THE CELL UNDER CONSIDERATION,
C        AND IF SO DETERMINE THE VALUE OF THE VECTOR HERE
C
         CALL GR_INSIDE ( IIN, XCOR, 4, ALIMITS(1), ALIMITS(2) )
C
         IF (IIN .EQ. 1) THEN
            FCOR(1) = ZP(ICELG2(2,ICELL))
            FCOR(2) = ZP(ICELG2(4,ICELL))
            FCOR(3) = ZP(ICELG2(6,ICELL))
            FCOR(4) = ZP(ICELG2(8,ICELL))
            CALL GR_CONTOUR_VALUE (XCOR, FCOR,
     1         ALIMITS(1), ALIMITS(2), UVAL)
            FCOR(1) = ZT(ICELG2(2,ICELL))
            FCOR(2) = ZT(ICELG2(4,ICELL))
            FCOR(3) = ZT(ICELG2(6,ICELL))
            FCOR(4) = ZT(ICELG2(8,ICELL))
            CALL GR_CONTOUR_VALUE (XCOR, FCOR,
     1         ALIMITS(1), ALIMITS(2), VVAL)
            GOTO 3020
         ENDIF
3010  CONTINUE
C
3020  AMAG = SQRT(UVAL**2 + VVAL**2)
      AANG = 0
      IF (AMAG .NE. 0) AANG = ATAN2D (VVAL, UVAL)
```

```
      WRITE (ISTRING,3030) UVAL, VVAL, AMAG, AANG
3030  FORMAT (' U=',G14.5,3X,'V=',G14.5,3X,'MAG=',G14.5,
     1        3X,'ANGLE=',F9.3,2X,'DG')
      RETURN
C
C     -------------
C     PLOT CONTOURS
C     -------------
C
4000  CONTINUE
C
C     GET THE REQUIRED SYMBOL SIZE
C
      CALL GR_GET_IOINF (DUM,DUM,DUM,LDEV)
      IF (LDEV .NE. 41) CALL PLTOFF
      SIZE = SQRT(FLOAT(NNODG2))*MAX(UMAX,VMAX)
      SIZE = 1./SIZE
      WRITE(JTERMO,*) ' RECOMMENDED VALUE OF SYMSIZ',SIZE
      CALL GR_REAL('ENTER SYMBOL SIZE',SYMSIZ)
      IF (LDEV .NE. 41) CALL PLTON
      SIZE = ABS(SYMSIZ)
C
C     SCALE SIZE TO SOMETHING RELATED TO THE DRAWING SCALE
C
      CALL GR_GET_SCALE (X1, X2, Y1, Y2)
      SZX  = SIZE*(X2-X1)
      SZY  = SIZE*(Y2-Y1)
      SIZE = MAX(SZX,SZY)
C
C     FOR EACH POINT, CALCULATE FLOW ANGLE AND MAGNITUDE
C     AND PLOT BODY OF ARROW.
C
      DO 4010 I = 1, MAXNOD
         J  = IPOINT(I)
         UU = ZP(J)
         VV = ZT(J)
         CALL GK_MOVE ( ZX(J), ZY(J), 0 )
         XX = SIZE*UU + ZX(J)
         YY = SIZE*VV + ZY(J)
         CALL GK_DRAW ( XX, YY, 0 )
C        DRAW THE HEAD OF THE ARROW
         IF (SYMSIZ .GE. 0.) THEN
            XX1 = XX + SIZE*(-.25*UU - .15*VV)
            YY1 = YY + SIZE*(-.25*VV + .15*UU)
            CALL GK_DRAW ( XX1, YY1, 0 )
            XX2 = XX + SIZE*(-.25*UU + .15*VV)
            YY2 = YY + SIZE*(-.25*VV - .15*UU)
            CALL GK_MOVE ( XX2, YY2, 0)
            CALL GK_DRAW ( XX,  YY,  0)
         ENDIF
4010  CONTINUE
      RETURN
C
5000  CONTINUE
C
C     RESET THE MARKED NODES IF PLOTTING MORE THAN ONCE
C
```

```
      DO  INODE = 1, NNODG2
         IMARKN(INODE) = 0
      ENDDO
C
      DO 5080 ICELL = 1, NCELG2

C        SET THE POINTERS FOR THIS CELL

         KC     = ICELG2(1,ICELL)
         KSW    = ICELG2(2,ICELL)
         KS     = ICELG2(3,ICELL)
         KSE    = ICELG2(4,ICELL)
         KE     = ICELG2(5,ICELL)
         KNE    = ICELG2(6,ICELL)
         KN     = ICELG2(7,ICELL)
         KNW    = ICELG2(8,ICELL)
         KW     = ICELG2(9,ICELL)
         KX     = KAUXG2(ICELL)
         K5LEVI = IAND(KX,KUOOOF)
         LEVELI = ISHFT(K5LEVI,-16)
C
C        MARK THE NODES WHICH ARE DONE WITH IMARKN(NODE)=-1
C
         IF (LEVELI .EQ. 0) THEN
            IF (IMARKN(KSW) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KSW)   = -1
               IPOINT(KOUNT) = KSW
            ENDIF
C
            IF (IMARKN(KSE) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KSE)   = -1
               IPOINT(KOUNT) = KSE
            ENDIF
C
            IF (IMARKN(KNE) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KNE)   = -1
               IPOINT(KOUNT) = KNE
            ENDIF
C
            IF (IMARKN(KNW) .NE. -1) THEN
               KOUNT         = KOUNT + 1
               IMARKN(KNW)   = -1
               IPOINT(KOUNT) = KNW
            ENDIF
C
         ENDIF
C
5080  CONTINUE
C
      MAXNOD = KOUNT
      GRDUMY(23) = MAXNOD
C
      RETURN
      END
```

D.6 Sample input files

This section contains the sample files INPUTI.DAT containing the input parameters and INPUTC.DAT containing the chemistry information for the reacting scramjet inlet problem in Section (8.3.2).

D.6.1 INPUTI.DAT

```
 2-D REACTING SCRAMJET WITH PREMIXED FUEL INJECTION; M=10
* COMMENT
ALPHA2=1.0
AMCHFL=6.6569
BETAA2=0.2
CFLNTI=0.5
CFLXTI=0.5
DELTA2=0.1
EPSOTI=0.01
EPS1TI=0.05
EPSLE2=1.E-10
ERRMTI=1.0
FCTRTI=1.0
GAMMA2=0.4
IMPLTI=    0            1: EXPLICIT
JREADS=    078
K1ADA2=    1
K2ADA2=    0
KADPTI=   99
KDPENI=    2
KEQNE2=    6
KFACTI=    0
KMERA2=    1
KONVE2=    2
KROGER=    1
KSRTE2= 1001
MALVG2=    2
METHA2=    6
MITRA2=10000
MITRE2= 1000
MITRPS=  201
NGIVTI=    2
MTHRA2=    1
NREACH=    2
NSPECH=    5
NXTDA2=    2
PRINTO=0.2
PRESFL=80000.
SMAXE2=0.50
SMINE2=0.05
TIMXTI=10000.
TREFFL=880.
```

```
TRIGCH=1000.
```

D.6.2 INPUTC.DAT

```
    0
 1  0.2276642  0.2276642
 2  0.0000000  0.0000000
 3  0.0229493  0.0229493
 4  0.0000000  0.0000000
 5  0.7493865  0.7493865
 1    1   0   1   0   0
 1    0   2   0   0   0
 2    0   2   1   0   0
 2    0   0   0   2   0
 1  5
      0.8        -10.      2448.4
      3.26439     0.       8992.0
 2  5
    -1500.        -13.     18940.6
     -19.7367      1.    -69415.0
```