NASA Contractor Report 3369

Hyperbolic/Parabolic Development for the GIM-STAR Code

L. W. Spradley, J. F. Stalnaker,
and A. W. Ratliff

CONTRACTS NAS1-15783 and NAS1-15795
DECEMBER 1980

NASA

NASA Contractor Report 3369

Hyperbolic/Parabolic Development for the GIM-STAR Code

L. W. Spradley, J. F. Stalnaker,
and A. W. Ratliff
Lockheed Missiles & Space Company, Inc.
Huntsville, Alabama

Prepared for
Langley Research Center
under Contracts NAS1-15783 and NAS1-15795

National Aeronautics
and Space Administration

Scientific and Technical
Information Branch

1980

CONTENTS

1. INTRODUCTION AND SUMMARY

The General Interpolants Method (GIM) code was developed to analyze complex flow fields which defy solution by simple methods. The code uses numerical difference techniques to solve the full three-dimensional time-averaged elliptic Navier-Stokes equations in arbitrary geometric domains. The equations are cast in strong Conservation law form and written in an orthogonal Cartesian coordinate system. Included are a continuity equation for global mass conservation, three components of momentum conservation, total energy conservation and an equation for conservation of individual species of a binary gas. Pressure is related to the conservation variables through the ideal gas law for a binary mixture. A generalized geometry package is used to model the flow domain, generate the numerical grid of discrete points and to compute the local transformation metrics. Computation is done in physical space by explicit finite-difference operators. The GIM approach essentially combines the finite element geometric point of departure with finite difference explicit computation analogs. This provides a capability which takes advantage of the geometric flexibility of an element description and the superior computation speed of difference representations.

The numerical analogs of the differential equations are derived by representing each flow variable with general interpolation functions. The point of departure then requires that a weighted integral of interpolants be zero over the flow domain. By choosing the weight functions to be the interpolants themselves, the GIM formulation produces identically the classical implicit finite element discrete equations. These forms are not used in the GIM code due to their fully implicit nature and inherent inefficiencies. Rather, the weight functions are chosen to be orthogonal to the interpolant functions which produces explicit finite difference type discrete analogs. By appropriate choice of constants in the weight functions, the GIM becomes analogous to such finite difference schemes as centered, backward, forward, windward and multi-step predictor-corrector schemes such as the MacCormack method. The GIM analogs, however, are automatically produced for arbitrary geometric flow domains and hence is a more general point of departure and provides greater flexibility in choosing difference schemes.

A motivation for developing this code on these principles was to provide an analytical tool which is more user oriented than the basic research tools which exist. A fully production-line code to solve the complex Navier-Stokes equations does not exist today. In developing the GIM code, an attempt was made to bridge the gap somewhat between the pure research codes and the ultimate production tool. The code was originally developed for a CDC 7600 computer system. It has been subsequently converted to vector FORTRAN for the CDC STAR-100 system at NASA Langley Research Center. Reference 1 provides documentation for the GIM/STAR code designated version SE-1 (STAR-Elliptic No. 1). This version of the code has been used to compute a number of complex flow fields including nozzle flows for both subsonic and supersonic regimes, and two and three-dimensional Scramjet exhaust flow simulations (Ref. 2).

The current contract work involves utilization and extensions of the GIM/STAR code. The objectives of the study are to:

- Compute flow fields in supersonic inlet configurations using the SE-1 code
- Upgrade the technical capability of the SE-1 code
- Develop a hyperbolic and a parabolic version of the GIM/STAR code to supplement the elliptic capability.

This report presents the progress to date on the development and application of the GIM/STAR code.

Section 2 presents the results of an application of the code to a two-dimensional supersonic inlet. The calculation was started upstream of the compression surface which turns at 25 deg to the horizontal. The Mach = 5 freestream flow generates a bow shock off the leading edge of the ramp. The calculation involved two primary considerations; (1) determine the amount of flow captured and the amount spilled into the freestream and (2) compute the inlet flow field and predict the shock wave/boundary layer interaction. The problem was run in two parts with the GIM code on the STAR machine. The ingested flow was determined (inviscidly) first and found to be 66% of the incoming stream. This agrees well with the numbers for which the simulated inlet was designed. The flowfield distribution at the nozzle entrance was then used to drive the internal flow allowing the performance parameters to be determined. The flow angularity produces a shock wave off the cowl lip which propagates into the nozzle. The ultimate interaction of this shock and the laminar boundary layer on the upper propulsion surface were computed. All shock waves were determined using the "capture" mode of calculation. Section 2 shows the computed solution for the spillage part of the flow and for the internal nozzle portion. The separation of the boundary layer due to the adverse pressure gradient is clear from the velocity and pressure contour plots. Radial distributions of the steady state flow field are given and a "time" history of the shock/boundary layer interaction calculation is also shown.

Section 3 of this report describes an investigation of linearized block implicit (LBI) finite difference schemes for the GIM code. The current explicit MacCormack schemes are relatively efficient for flows with inviscid boundary conditions. In anticipation of other requirements to compute three-dimensional viscous flows, the necessity of eliminating the explicit stability limit becomes apparent. However, the extreme inefficiencies inherent in "fully" implicit methods, due to the large band-width matrices, make them unrealistic for large three-dimensional viscous flow problems. The most promising concept is the linearized block implicit (LBI), or approximate factorization, schemes. These methods retain the Conservation Law equation form while "splitting" the spatial dependence in the manner of the ADI schemes. The resulting matrix bandwidth is once again small (usually 3) and is practical to use. The study of LBI schemes in this work was concentrated on:

- Stability requirements of the block tridiagonal scheme of Beam-Warming (Ref. 3)
- Accuracy of the LBI scheme itself and more precisely, the accuracy and speed of linear equation solvers for vector machines
- Shock wave resolution of LBI schemes used in a capture mode and artificial damping requirements
- Techniques to vectorize LBI schemes for use on the STAR-100 machine.

The study was carried out with a one-dimensional code that uses the Beam-Warming formulation.

Results of the LBI investigation are discussed in detail in Section 3. The stability of the scheme was found to be strongly coupled to the accuracy of the linear equation solver used and to the artificial damping added to the explicit side of the scheme. The "unconditional" stability indicated by the theory could not be achieved numerically using centered differences. Schemes based on one-sided windward differences did prove to be unconditionally stable. The LBI scheme was shown to be as good as the explicit MacCormack for resolution of shock waves. The overall conclusion of this part of the study is that LBI schemes appear to be very promising for three-dimensional viscous flows but they are not as outstanding as the literature indicates.

The third part of this study reported here is the development of hyperbolic and parabolic methods to supplement the elliptic code. Section 4 describes the details of the work on the GIM maching algorithms and the current status of the code. The basic idea of the GIM code marching scheme is to combine the classical parabolized Navier-Stokes methods with a "quasi-time" relaxation. The term "quasi-parabolic" (QP) will be used to refer to this algorithm although the scheme applies equally well to hyperbolic, supersonic inviscid flows. The QP algorithm is contrasted to a fully elliptic method in that downstream effects cannot be felt upstream and that a full flow domain need not be stored for the QP scheme. The QP algorithm is also contrasted to classical parabolic methods in that mixed subsonic/supersonic flows do not produce a multiple "decode" root and that real-wall no-slip boundaries can be treated with the QP algorithm. The equations are the classical parabolized Navier-Stokes but with a psuedo-time derivative added back to them. The solution is known at upstream data planes 1, 2, ... N-1 and the solution is sought at plane N with no influence from plane N+1. Time relaxation is used to solve for plane N from only the (converged) solution at upstream planes. Backward differences (second order) are used, of course, in the quasi-marching coordinate. As the algorithm is formulated, either explicit or linear block implicit time relaxation can be incorporated.

The resulting algorithm then requires much less computer storage than a GIM elliptic flow field calculation and does not have the "singularities" inherent in classical parabolic marching algorithms. The QP scheme has been coded and partially checked out on the STAR system. At the time of this writing, the GEOMETRY, MATRIX and INTEG modules of the SP-1 GIM code (STAR Parabolic, Version 1) have been run successfully for several sample cases.

Some details of the current contract work are appended. The most current version of the GIM elliptic code (SE-2) is discussed in Appendix A authored by L. W. Spradley. Differences in SE-1 and SE-2 are described and reasons for the changes explained. New INPUT data sheets for SE-2 are given to replace the ones in the "Blue Book" (Ref. 1). This basic guide should be used in conjunction with the Blue Book for inputting the GIM code on STAR. Appendix B by Jürgen Thoenes, contains a derivation and list of the complex linearization Jacobians for three-dimensional LBI schemes. The GIM-Marching code (SP-1) requires a special set of weight/shape functions. These are derived in Appendix C, which is authored by John F. Stalnaker. The final item to be covered here is a description of the vectorized linear algebraic equation solvers which were developed on the STAR system for use with the LBI schemes. The mathematical development and performance of several techniques, both direct and iterative are shown in Appendix D, authored by S. J. Robertson.

2. CALCULATION OF TWO-DIMENSIONAL INLET FLOWS WITH SPILLAGE

2.1 INTRODUCTION

Figure 2-1 shows the model two-dimensional supersonic inlet for which the flow field was computed using the elliptic GIM/STAR code. The compression surface makes a sharp 25 deg turn at $x = 0$. It turns 50 deg through a circular arc centered about $x = 5$ into the 25 deg expansion surface. The expression surface and the lower cowl from the nozzle. The freestream flight conditions are also shown in Fig. 2-1. All flor variables are made dimensionless with the freestream quantities.

For inlets with fixed geometry it is important to know the amount of flow captured by the inlet and the amount that spills into the freestream. Thus, special emphasis was placed on calculating the mass flow rate at the inlet throat ($x = 5$). The model inlet was designed inviscidly to capture 66.6% of the incident flow.

It is felt that a brief history of the development and an outline of the pitfalls incurred in obtaining the final solution would be of benefit to future users of the GIM/STAR code. This discussion appears in Section 2.2. A complete analysis of the final solution is given in Section 2.3. These discussions are divided into two parts: (1) the external flow field below the compression surface and including the freestream flow which spills below the cowl, and (2) the internal (nozzle) flow field.

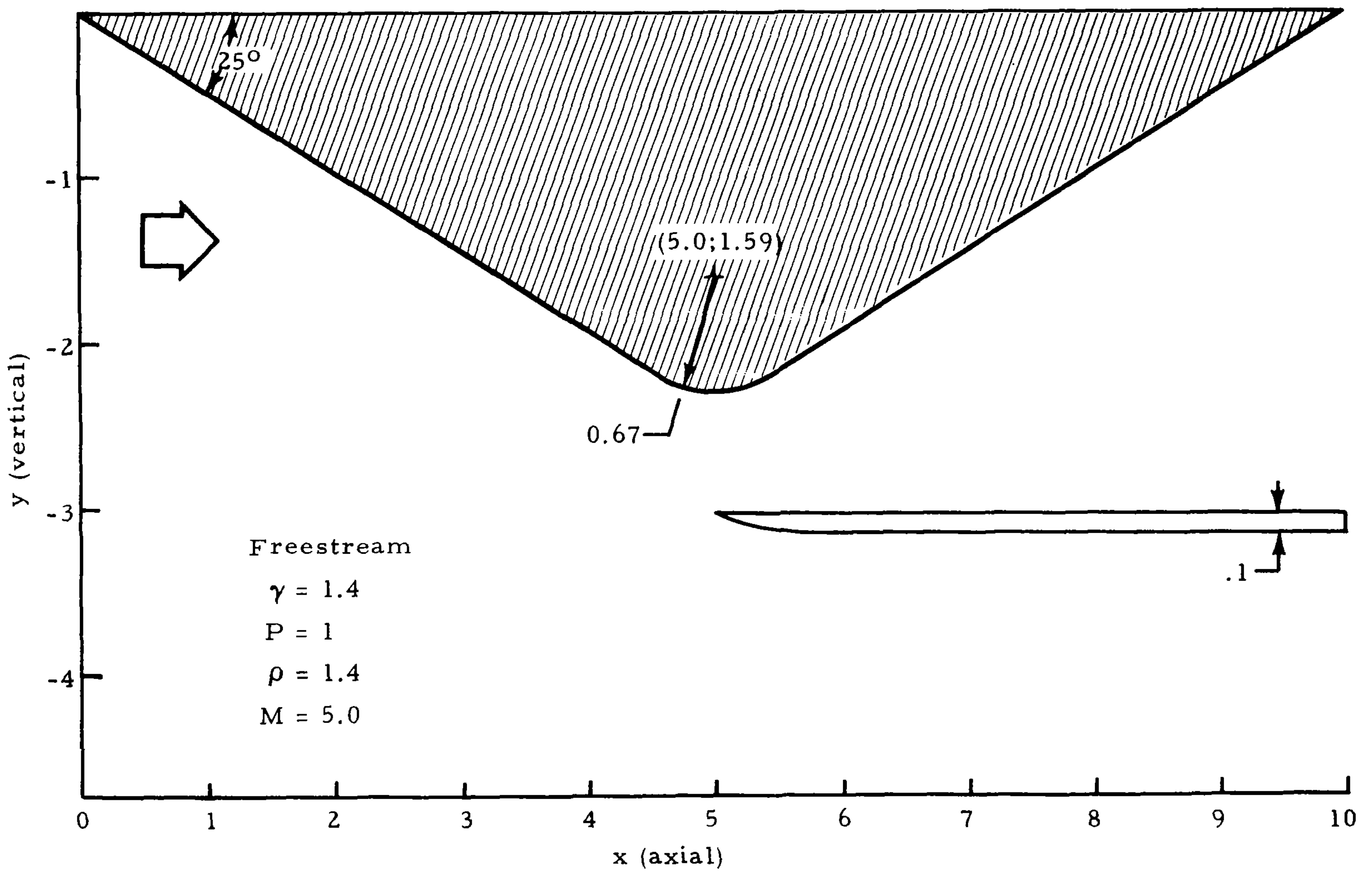


Fig. 2-1 - The Model Two-Dimensional Inlet

2.2 DEVELOPMENT OF THE SOLUTION

2.2.1 External Flow Field

To limit the problem size, the computational grid was constructed originally with the input boundary lying along the 35.7 degree bow shock line. This resulted in computational difficulties with the grid points along a horizontal line from the leading edge of the cowl to the shock line. In order to wrap the grid around the cowl, a discontinuity in the shapes of the elements arose along this line. Relatively uniform rectangular elements were mated to severely skewed elements. It is believed that the computational problems arose from the finite difference analogs generated along this line of nodes. These improper influences were caused by either sharp discontinuities in the transformation metrics or inadvertent extrapolation in the transformations.

As a result, the post-shock grid was abandoned and it was decided that the geometry should be constructed to allow the bow shock to be captured. The analogs for this grid were thoroughly examined using a coarsely spaced version of the final computational mesh (shown in Fig. 2-2). A "double-valued" node (i.e., two nodes at the same spatial location) was used to allow the proper splitting of the flow at the cowl lip. Due to the small shock angle at the bow which would not permit a sufficient number of nodes between the shock and the surface, the first eight nodes along the compression surface were held fixed at the inviscid post-shock conditions. This eliminated numerical disturbances which were otherwise generated at the bow and propagated downstream leading to instabilities.

2.2.2 Internal Flow Field

As originally modeled, the upper body of the inlet had a sharp 50 deg expansion at $x = 5$. In the initial inviscid analysis of the nozzle it was found that the flow overexpanded around this turn leading to pressure undershoot and instability. The sharp turn was rounded to alleviate this overexpansion. However, subsequent analysis revealed the problem to be excessive damping on the continuity equation. This resulted in an artificial dissipation

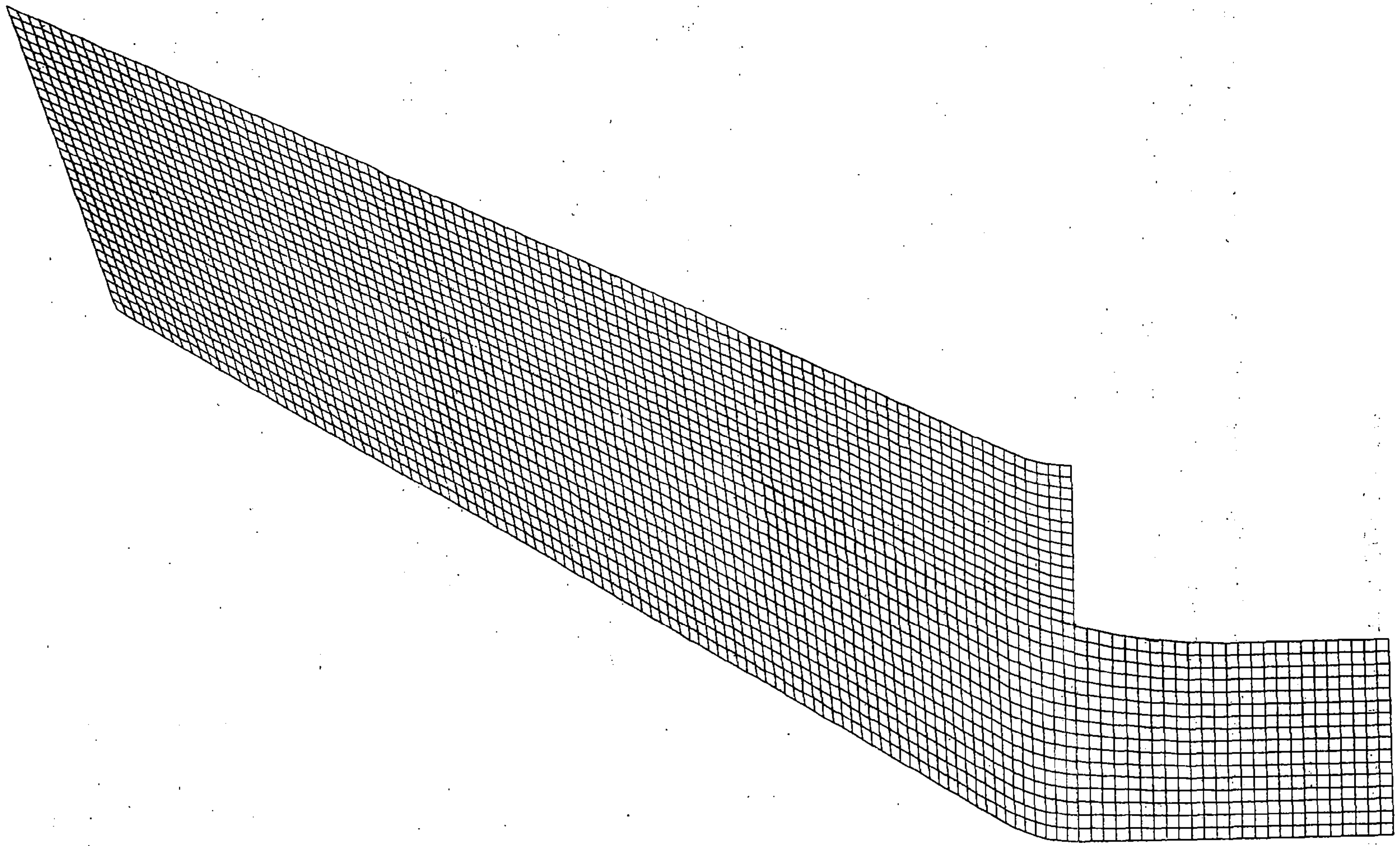


Fig. 2-2 - Computational Grid for External Flow in the Two-Dimensional Inlet

of mass away from the wall. Reduction of this damping allowed successful computation of the expansion; however, the rounded surface remained. The primary difficulty with the nozzle calculation was with the inviscid treatment of the expansion surface. This was first indicated by the failure of the inviscid SEAGULL code (Ref. 4) to converge in the nozzle. Imposing a viscous boundary layer on the upper wall allowed the GIM code to develop a strong shock-boundary layer interaction which made evident the fallacy in the inviscid treatment.

In arriving at the final solution it has become increasingly clear that solutions with the GIM code are strongly dependent on two factors: (1) the structure of the computational mesh, and (2) proper modeling of the physics of the problem.

2.3 RESULTS AND DISCUSSION

2.3.1 External Flow Field

The 3557 node computational grid for the external flow field is shown in Fig. 2-2. The solid boundaries were treated inviscidly. The USERIP option in the GIM/STAR code was used to initialize the flow field in order to lay in the bow shock as closely as possible to the inviscid 35.7 degree line. The solution converged to steady state in 900 iterations. The integrated mass flow rate indicated that the inlet captured 66.5% of the incident mass flow which compared almost exactly to the theoretical value. Figures 2-3 through 2-5 show the velocity vectors, pressure and Mach number contours for the complete flow field. Figure 2-6 shows a comparison of the mass flow rate ($m = \rho u$) across the inlet plane as calculated by the GIM/STAR code to that calculated by the inviscid SEAGULL code (Ref. 4) for a similar inlet with the same imposed capture rate. The agreement is excellent with the only apparent differences resulting from the different treatment of shocks in the two codes. For computational economy the deteched shock effects at the cowl lip were not treated here. Rather, after 100 iterations the values of the flow variables at the lip node were held fixed at attached post-shock conditions determined

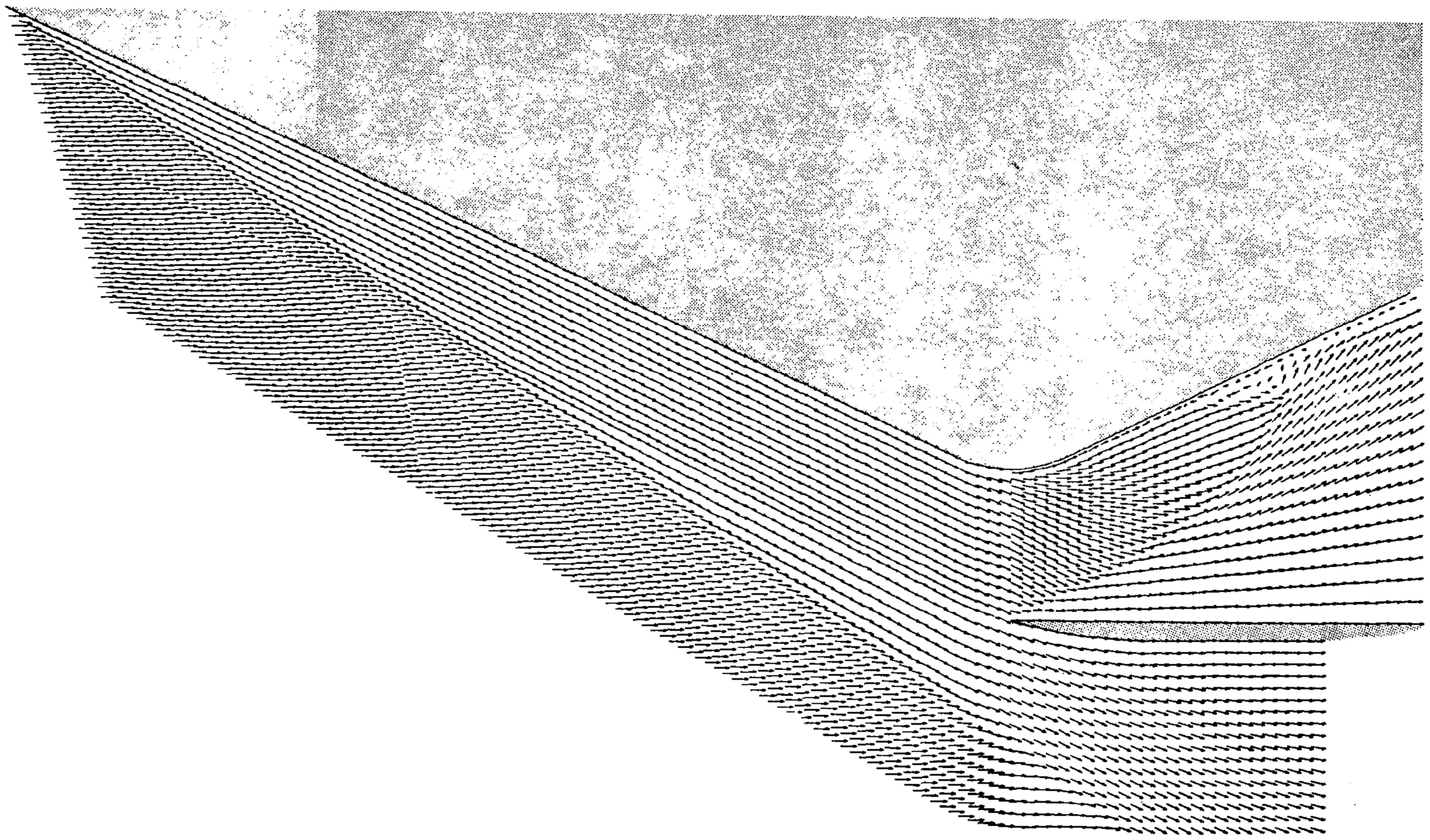
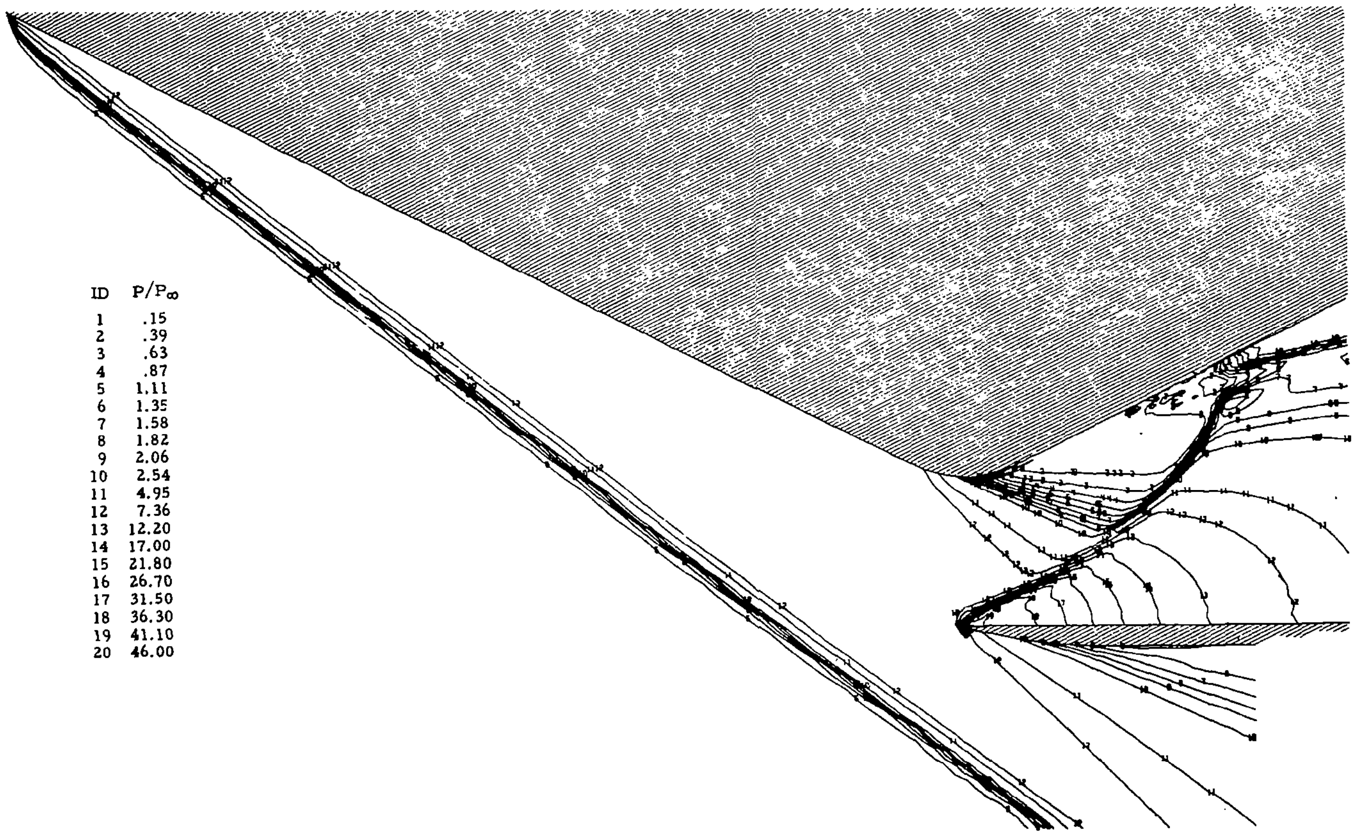


Fig. 2-3 - Velocity Vectors for the Two-Dimensional Inlet (Maximum Velocity = 5.0).



ID	P/P_∞
1	.15
2	.39
3	.63
4	.87
5	1.11
6	1.35
7	1.58
8	1.82
9	2.06
10	2.54
11	4.95
12	7.36
13	12.20
14	17.00
15	21.80
16	26.70
17	31.50
18	36.30
19	41.10
20	46.00

Fig. 2-4 - Pressure Contours for the Two-Dimensional Inlet.

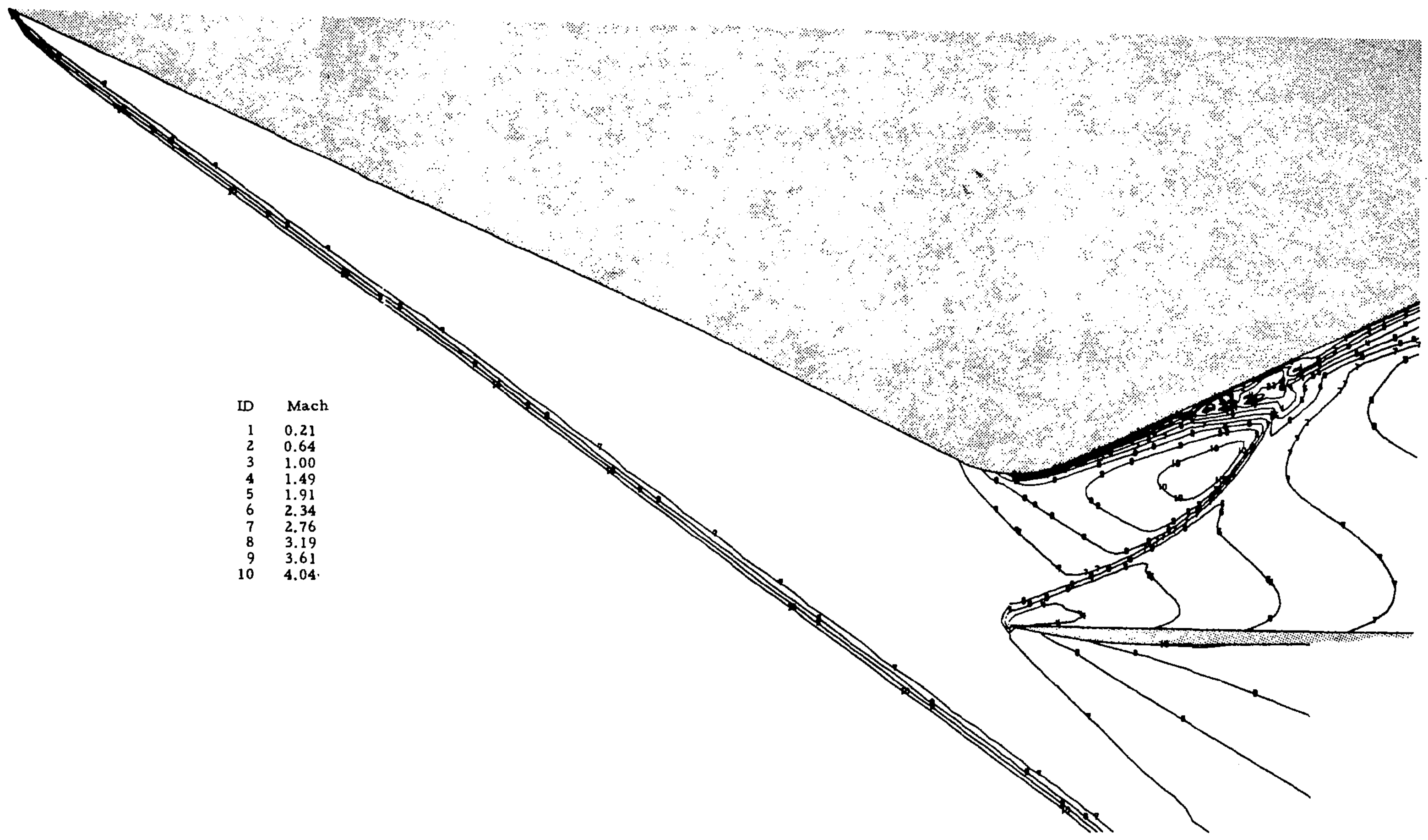


Fig. 2-5 - Mach Number Contours for the Two-Dimensional Inlet.

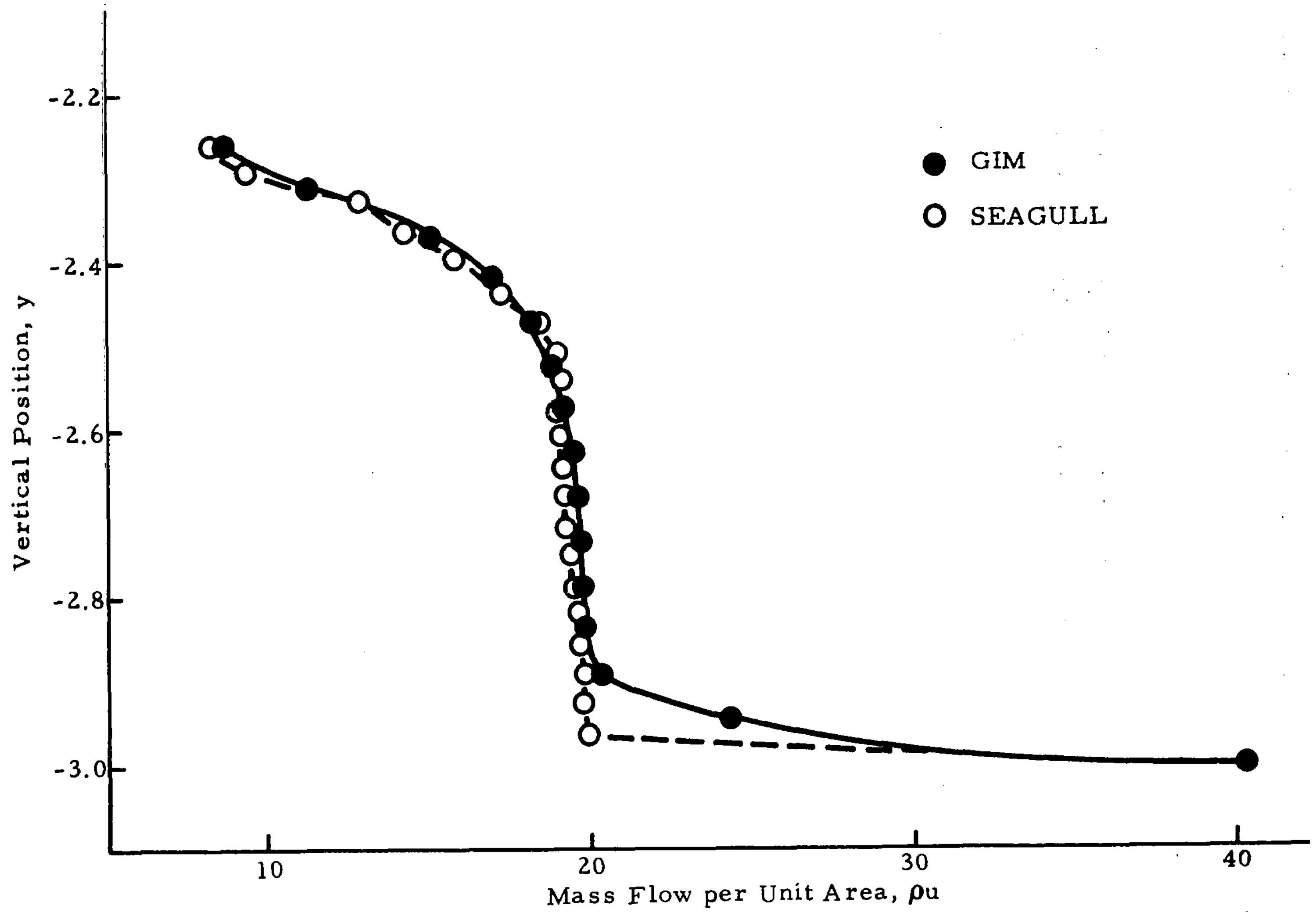


Fig. 2-6 - Mass Flow per Unit Area vs Vertical Position Across the Inlet Plane – (Station 5) Comparison with SEAGULL Solution (Ref. 4).

from conditions immediately upstream of the cowl lip. Further, when the lip shock was allowed to detach, the blunt body effects did not extend beyond one grid width from the lip.

2.3.2 Internal Flow Field

As noted in Section 2.2.2, it was necessary to impose a viscous boundary layer on the upper wall of the nozzle to obtain the solution. The boundary layer profile at the throat was estimated by a quadratic laminar profile with a Reynolds number of 1×10^4. The inlet flow variables were input by linear interpolation of the external flow results and the remaining nodes exterior to the boundary layer were initialized via the USERIP option by an isentropic-like area expansion along streamwise rows of nodes. It was found that allowing the code to develop the shock in the nozzle was preferable to estimating the shock position as was done in the external flow calculation. Figure 2-7 shows the 3000 node computational mesh for the nozzle. The solution converged to steady state in 1200 iterations. The final velocity vectors, pressure and Mach contours are shown in Figs. 2-3 through 2-5. Figures 2-8 through 2-20 show the time development of the solution from iteration 0 through 1200. Figures 2-21 through 2-24 show variations of the Mach number and pressure in the nozzle compared with the available SEAGULL results. It is apparent from these last figures that the boundary layer is artifically too thick (a result of the choice of Reynolds number). However, the result obtained provides considerable insight into the physics of the problem as well as the reasons behind the failure of the inviscid analysis.

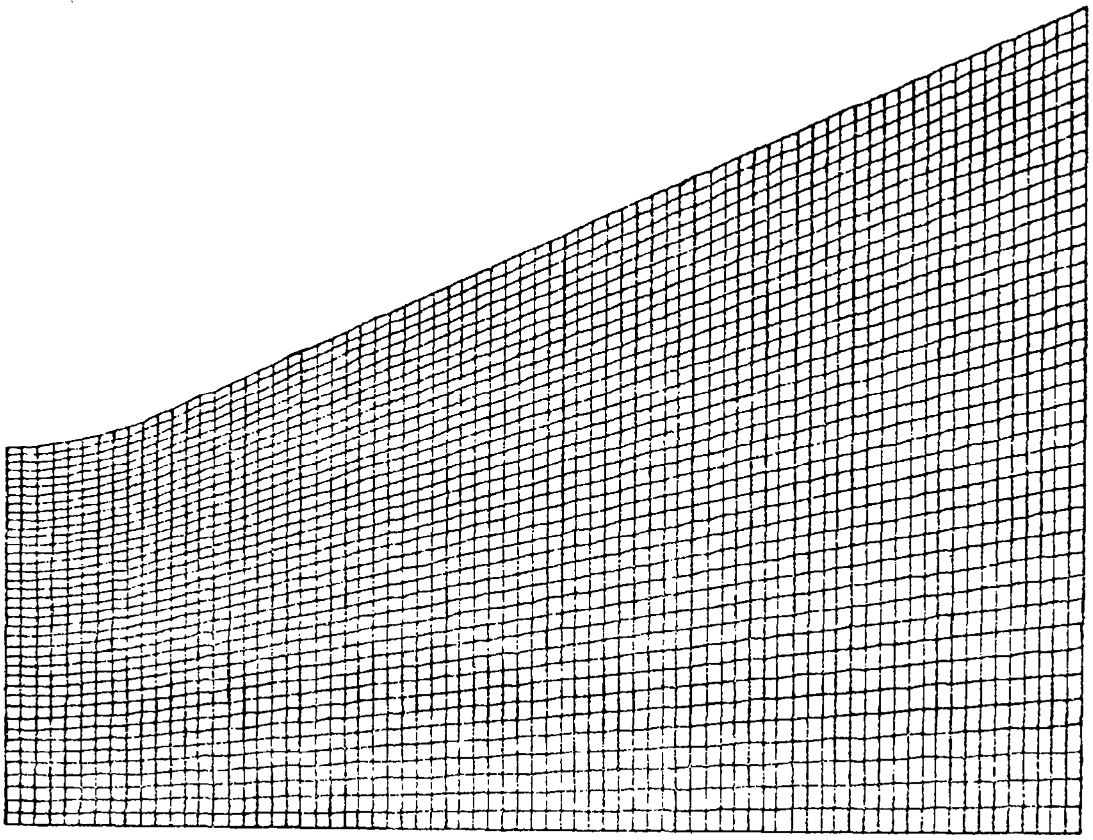


Fig. 2-7 - Computational Grid for Internal Flow in the Two-Dimensional Inlet.

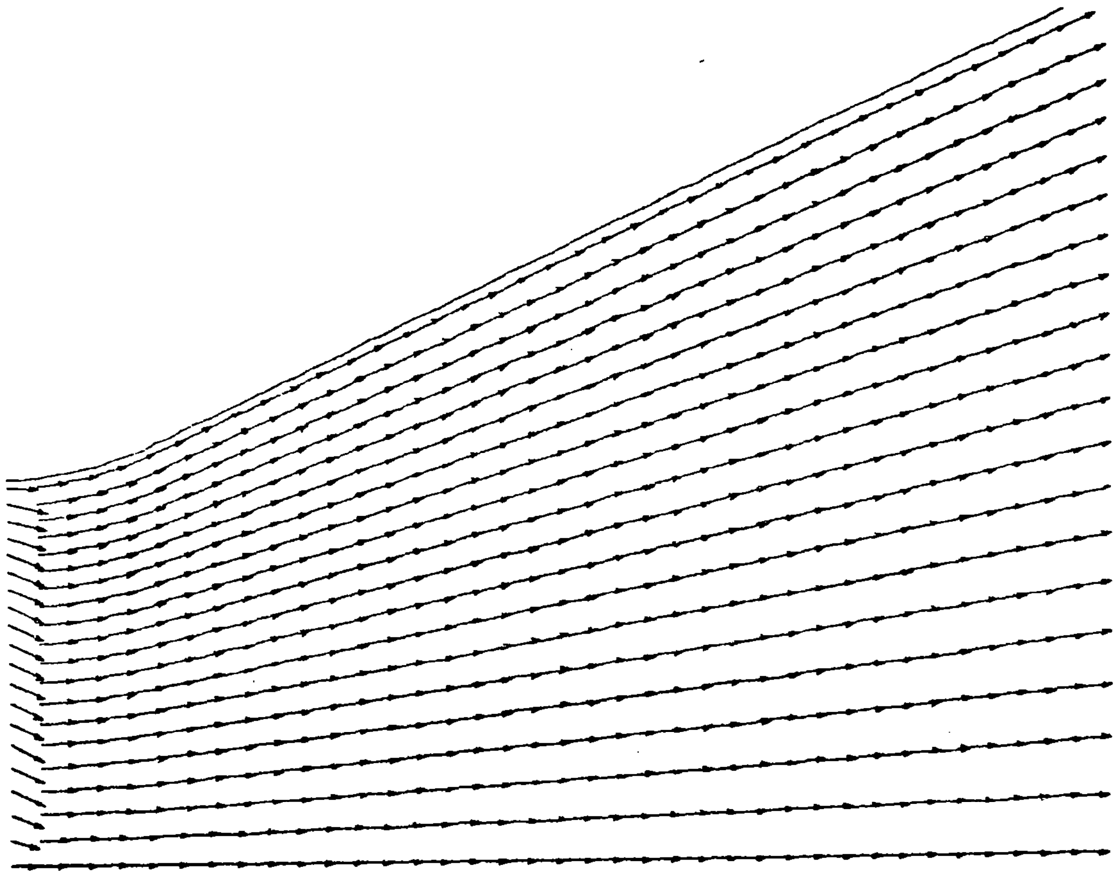


Fig. 2-8 - Nozzle Velocity Vectors (V_{max} = 4.91; No Iterations).

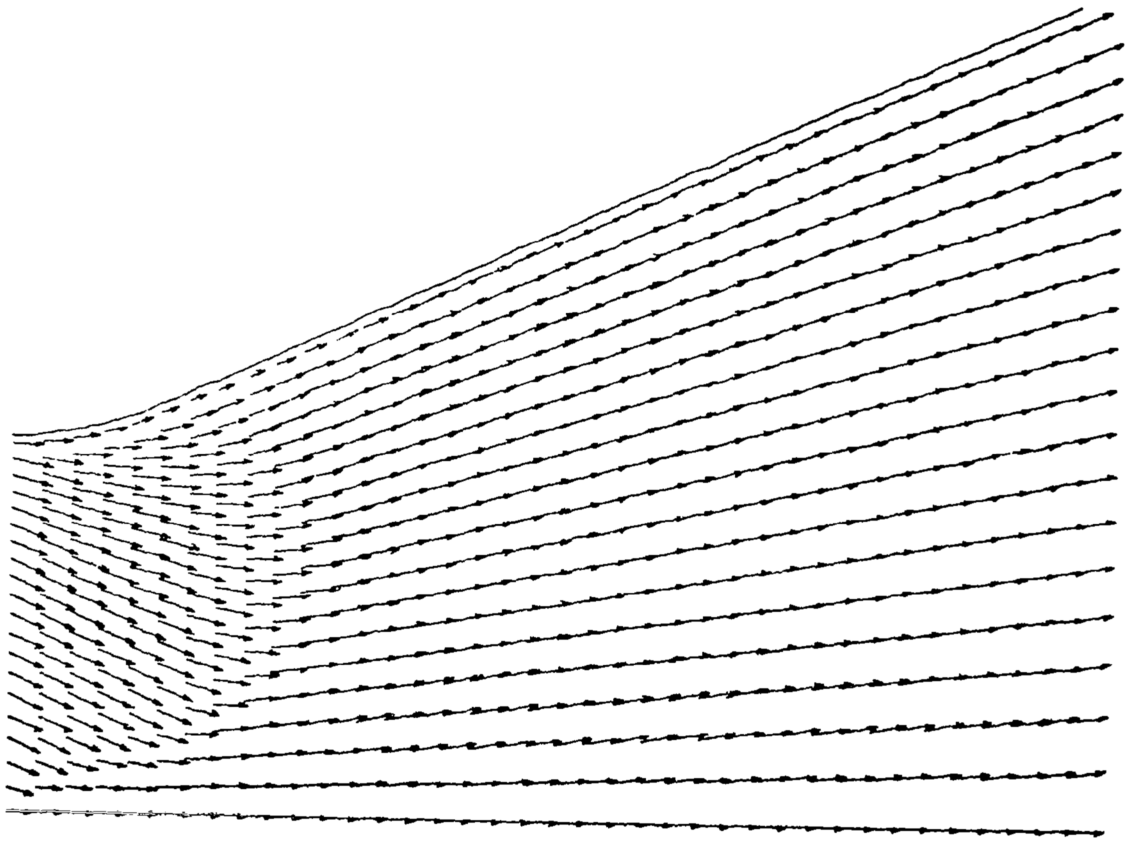


Fig. 2-9 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.92$; Iteration 100).

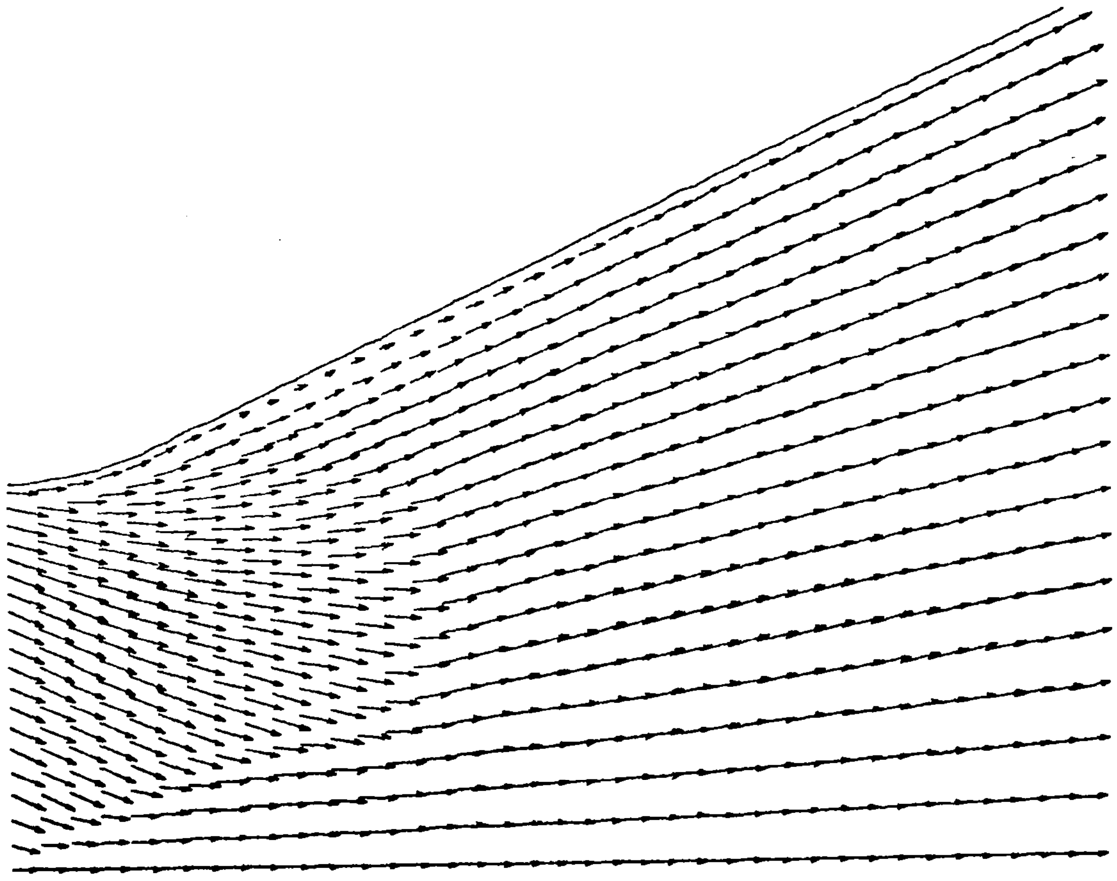


Fig. 2-10 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.98$; Iteration 200).

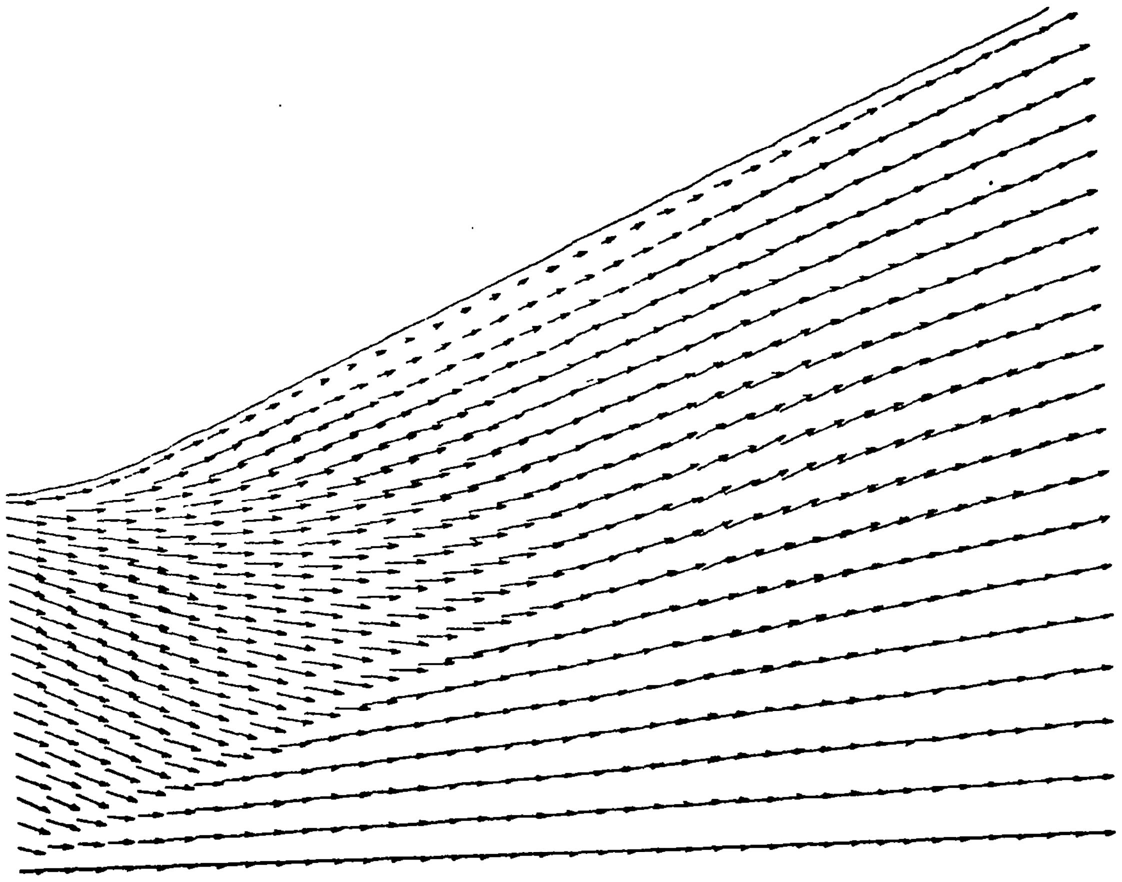


Fig. 2-11 - Two-Dimensional Spillage Problem (Viscous Nozzle; V_{max} = 5.00; Iteration 300).

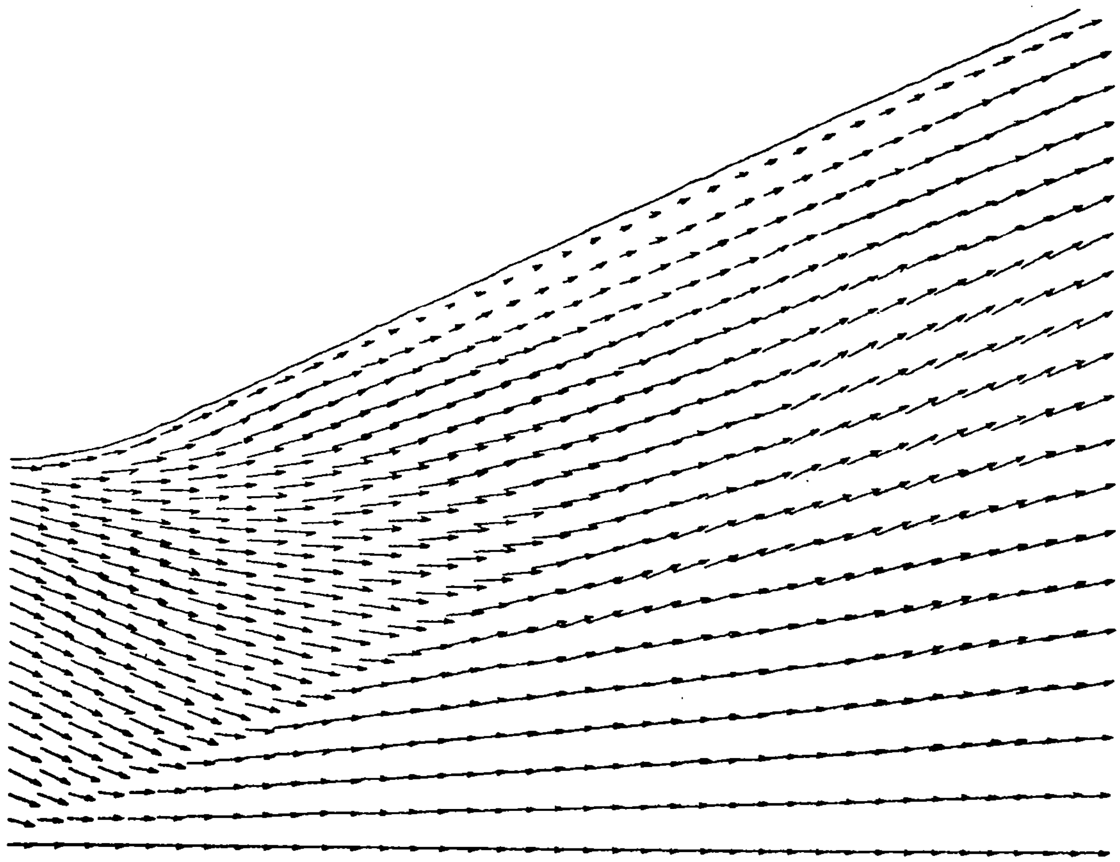


Fig. 2-12 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.84$; Iteration 400).

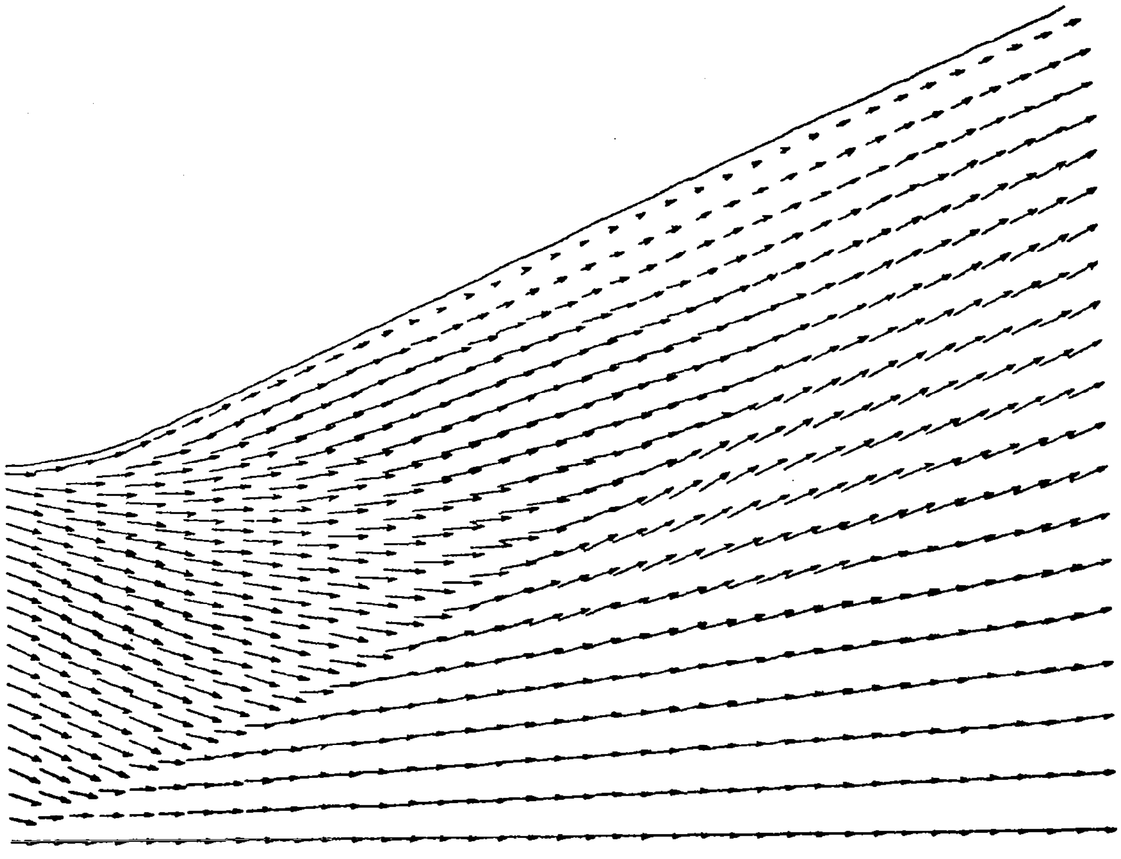


Fig. 2-13 - Two-Dimensional Spillage Problem (Viscous Nozzle; V_{max} = 4.88; Iteration 500).

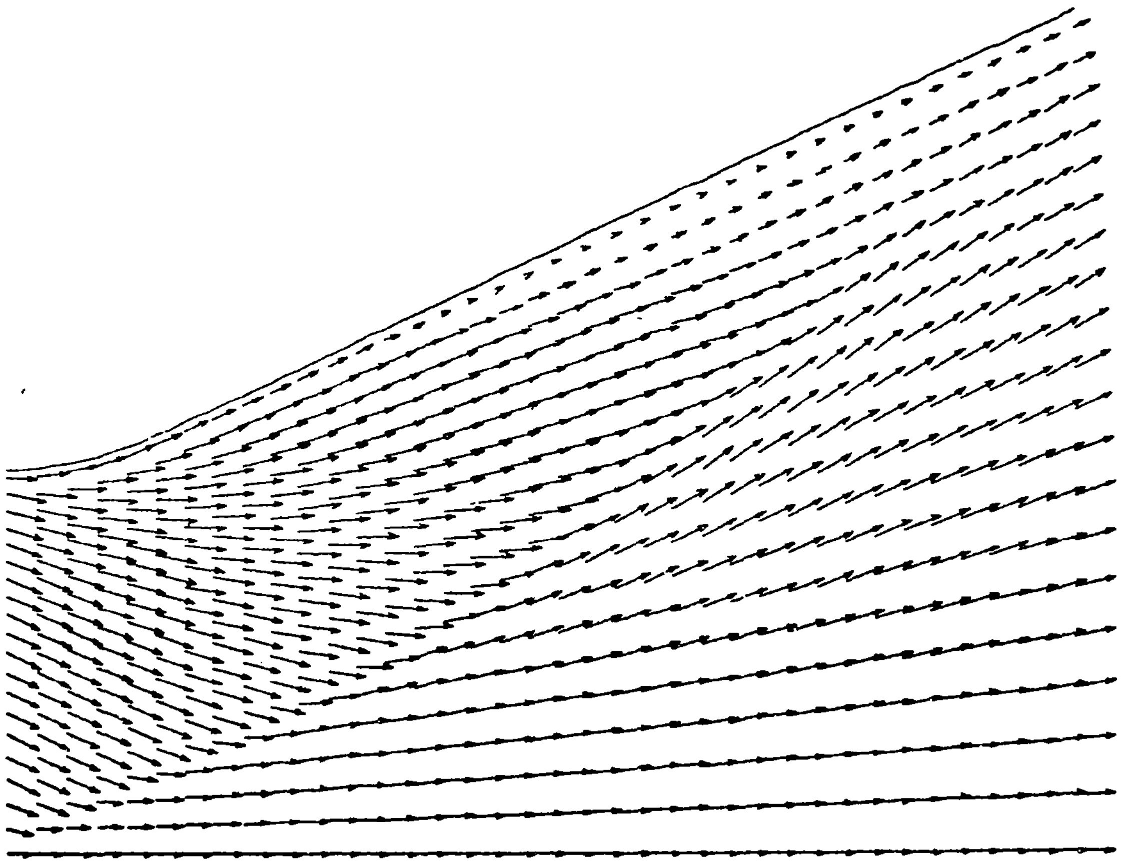


Fig. 2-14 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.82$; Iteration 600).

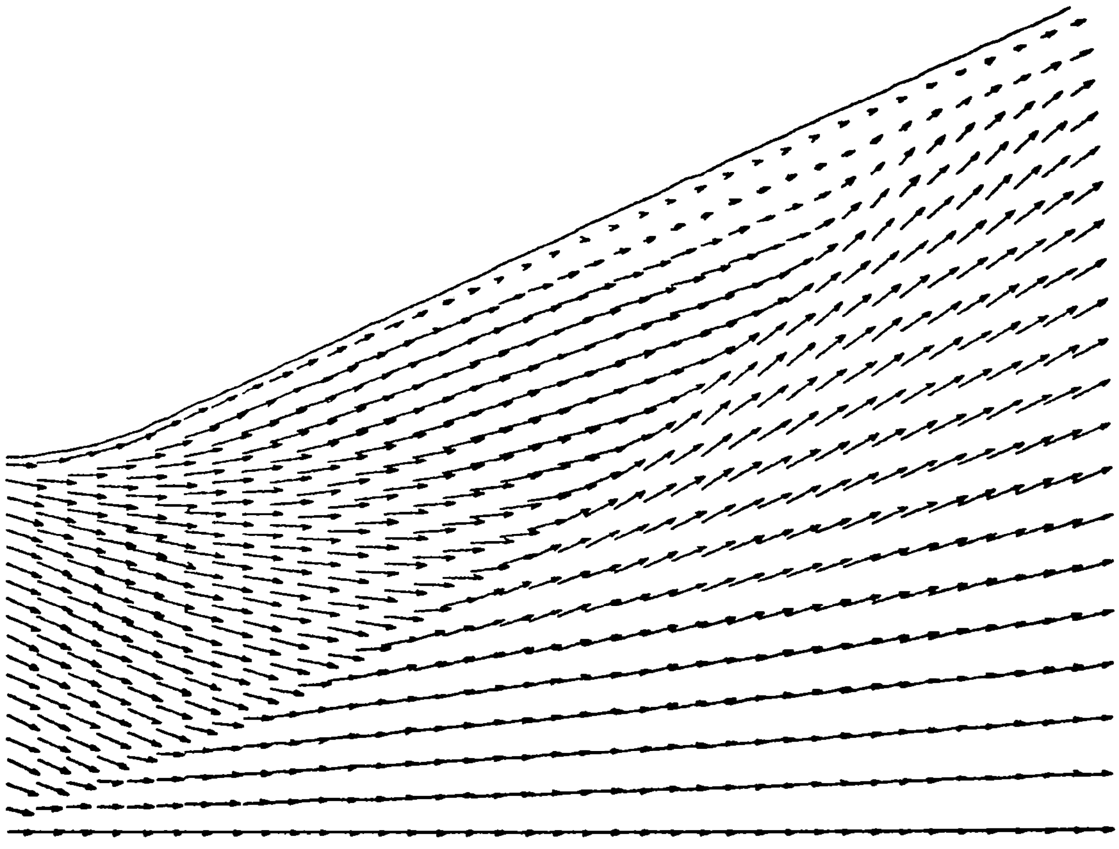


Fig. 2-15 - Two-Dimensional Spillage Problem (Viscous Nozzle; V_{max} = 4.87; Iteration 700).

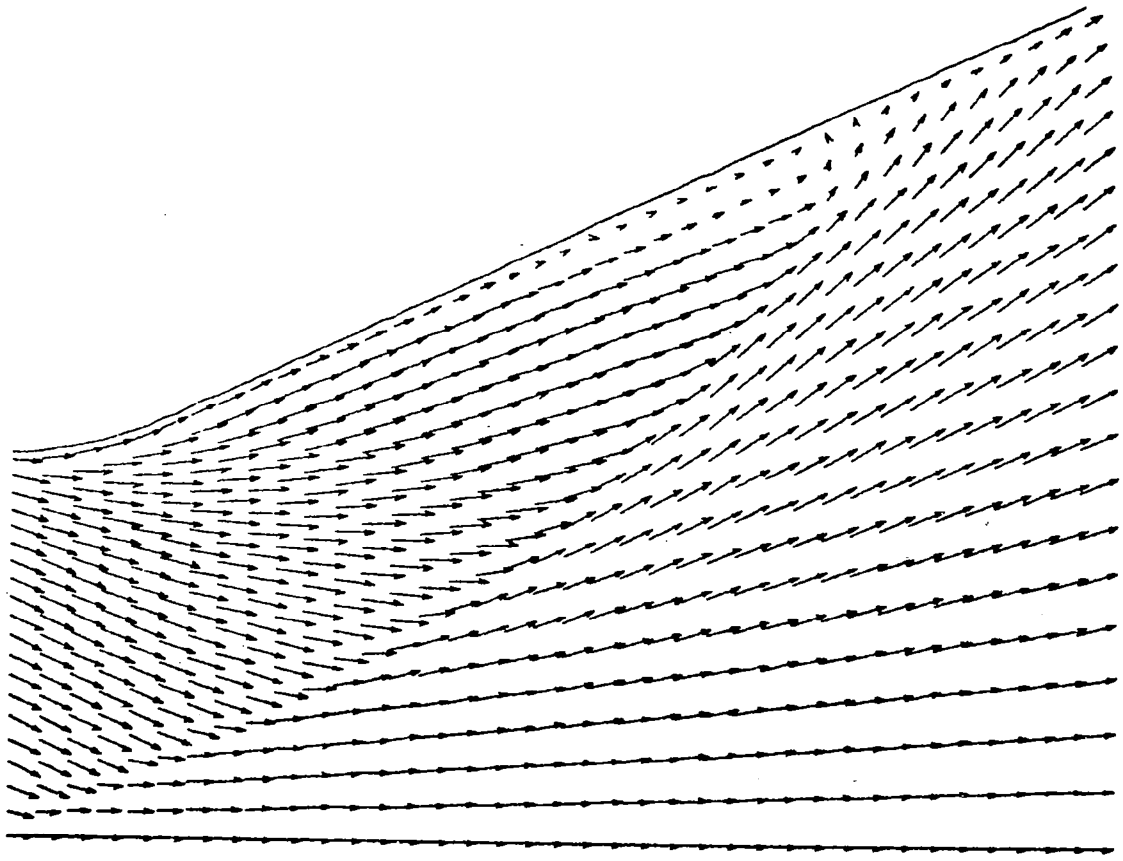


Fig. 2-16 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.90$; Iteration 800).

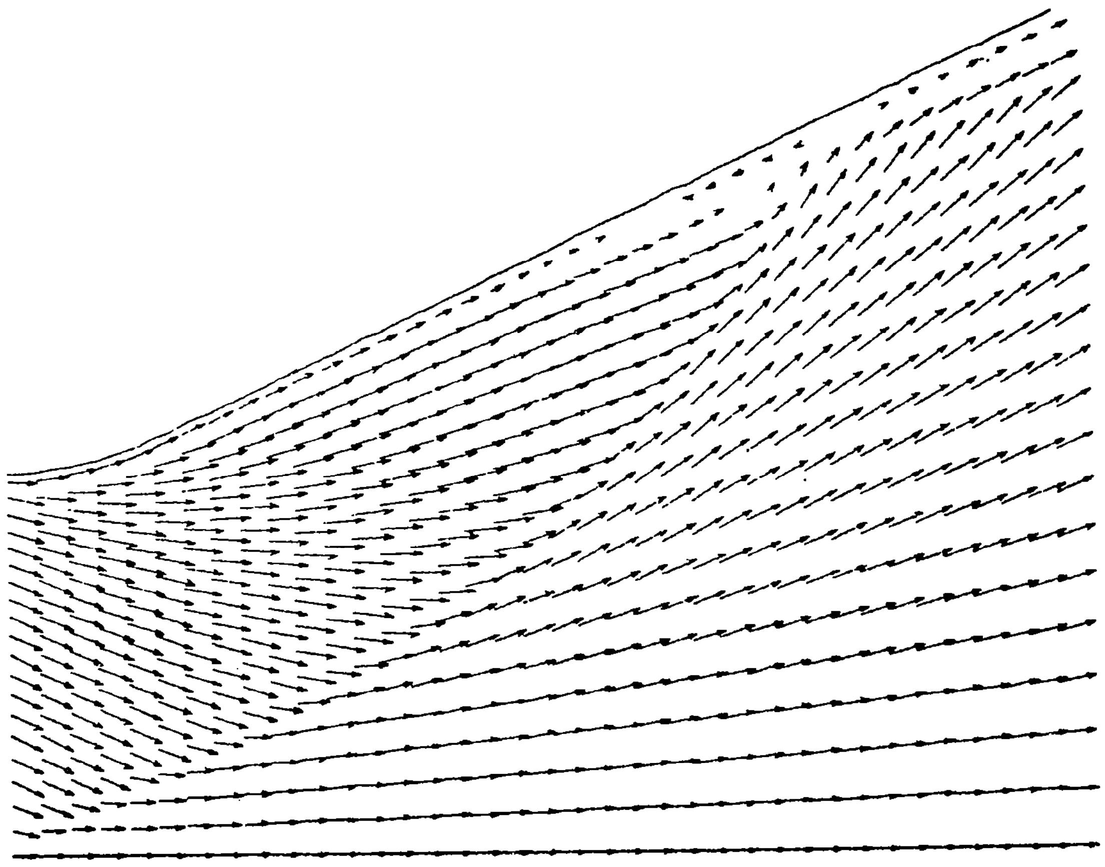


Fig.2-17 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.91$; Iteration 900).

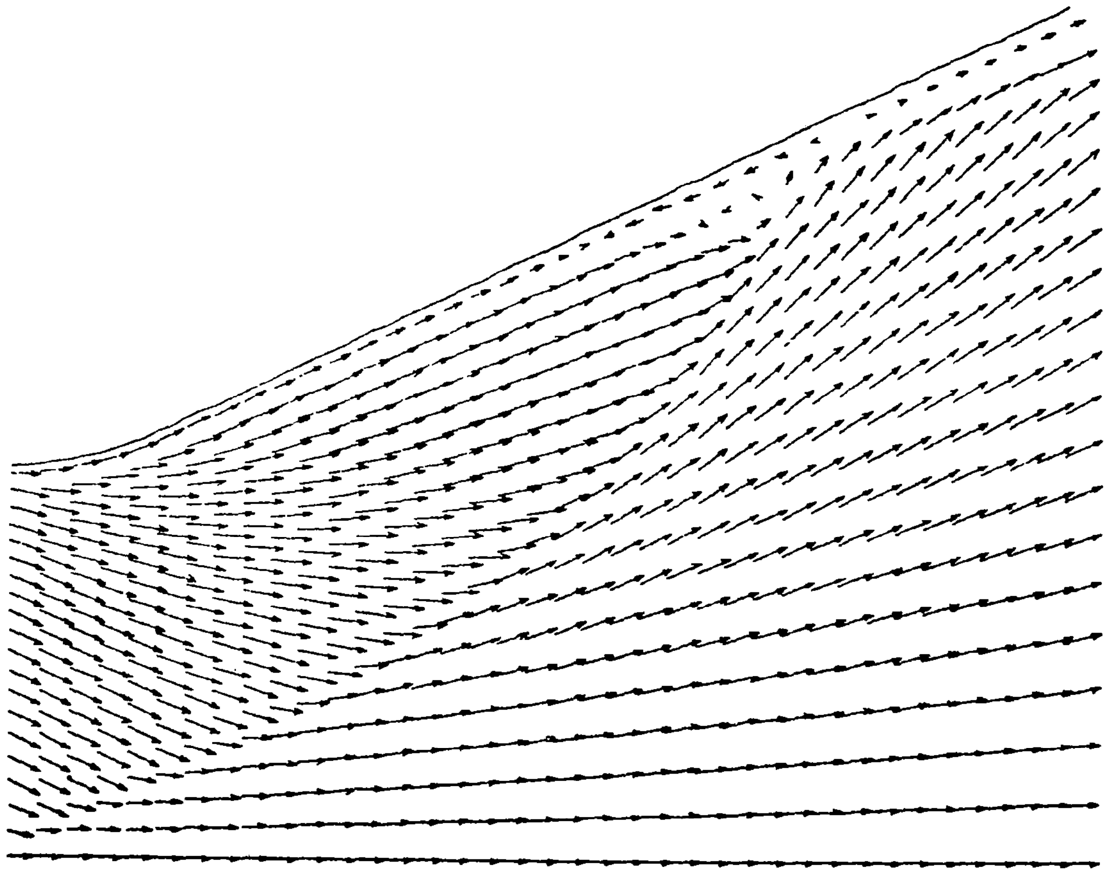


Fig. 2-18 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.91$; Iteration 1000).

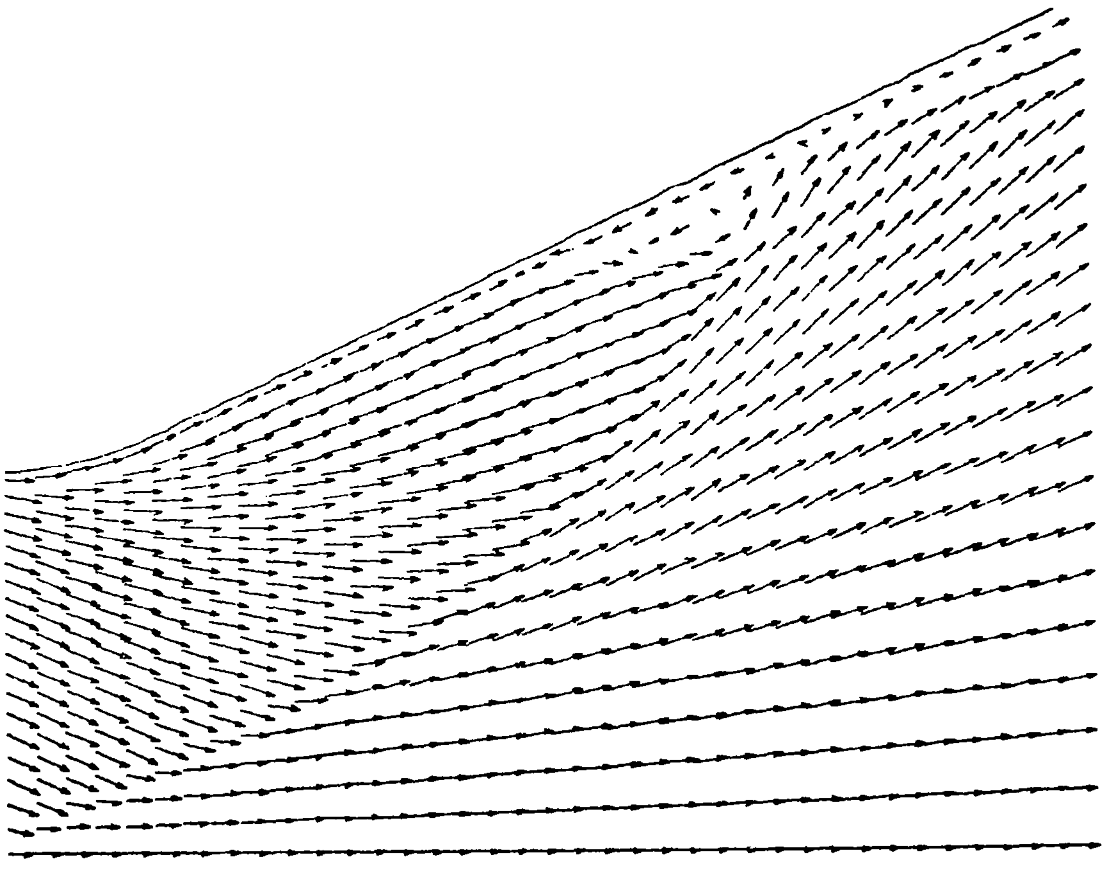


Fig. 2-19 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.91$; Iteration 1100).

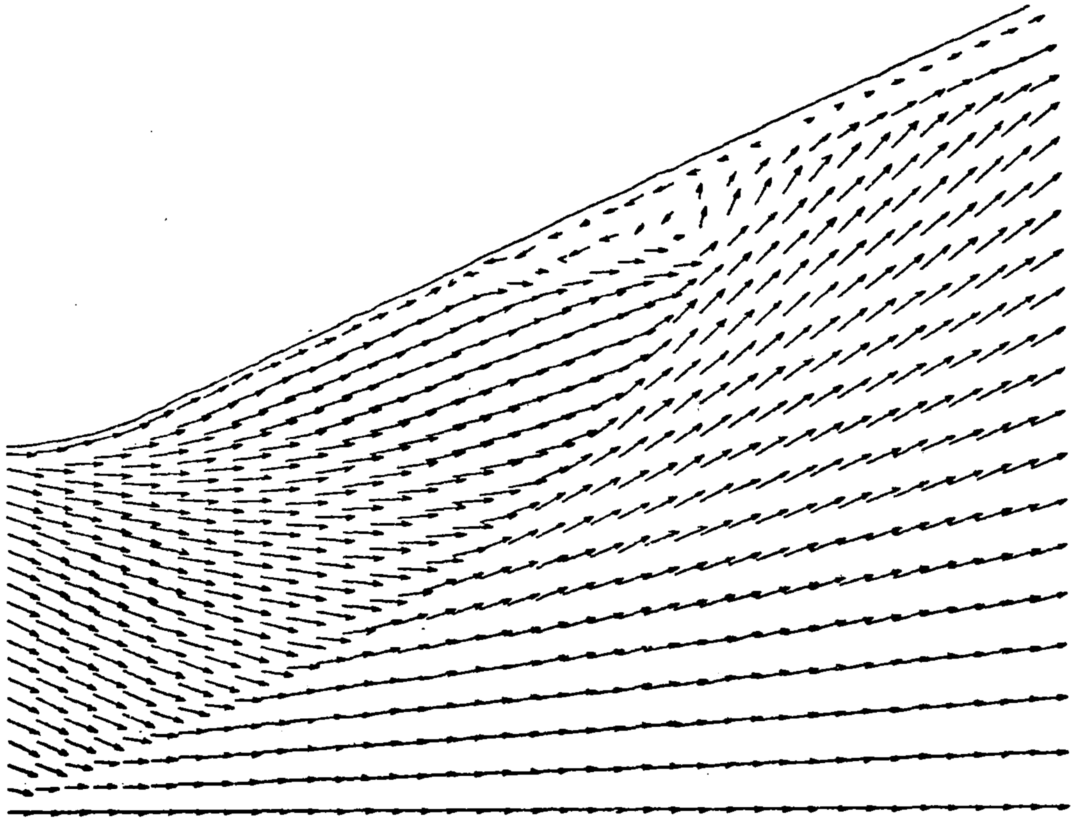


Fig. 2-20 - Two-Dimensional Spillage Problem (Viscous Nozzle; $V_{max} = 4.91$; Iteration 1200).

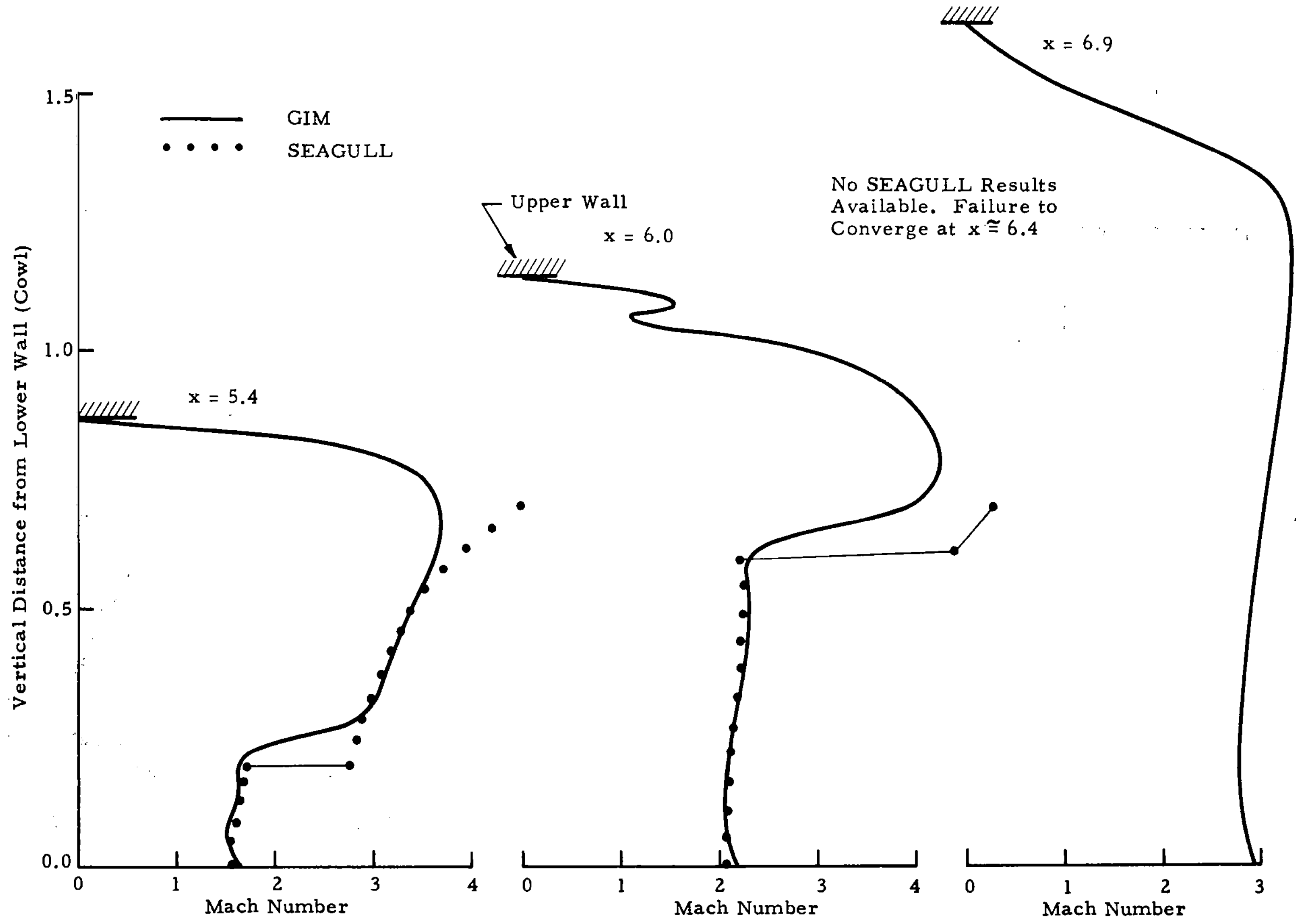


Fig. 2-21 - Mach Number Variations Across the Nozzle.

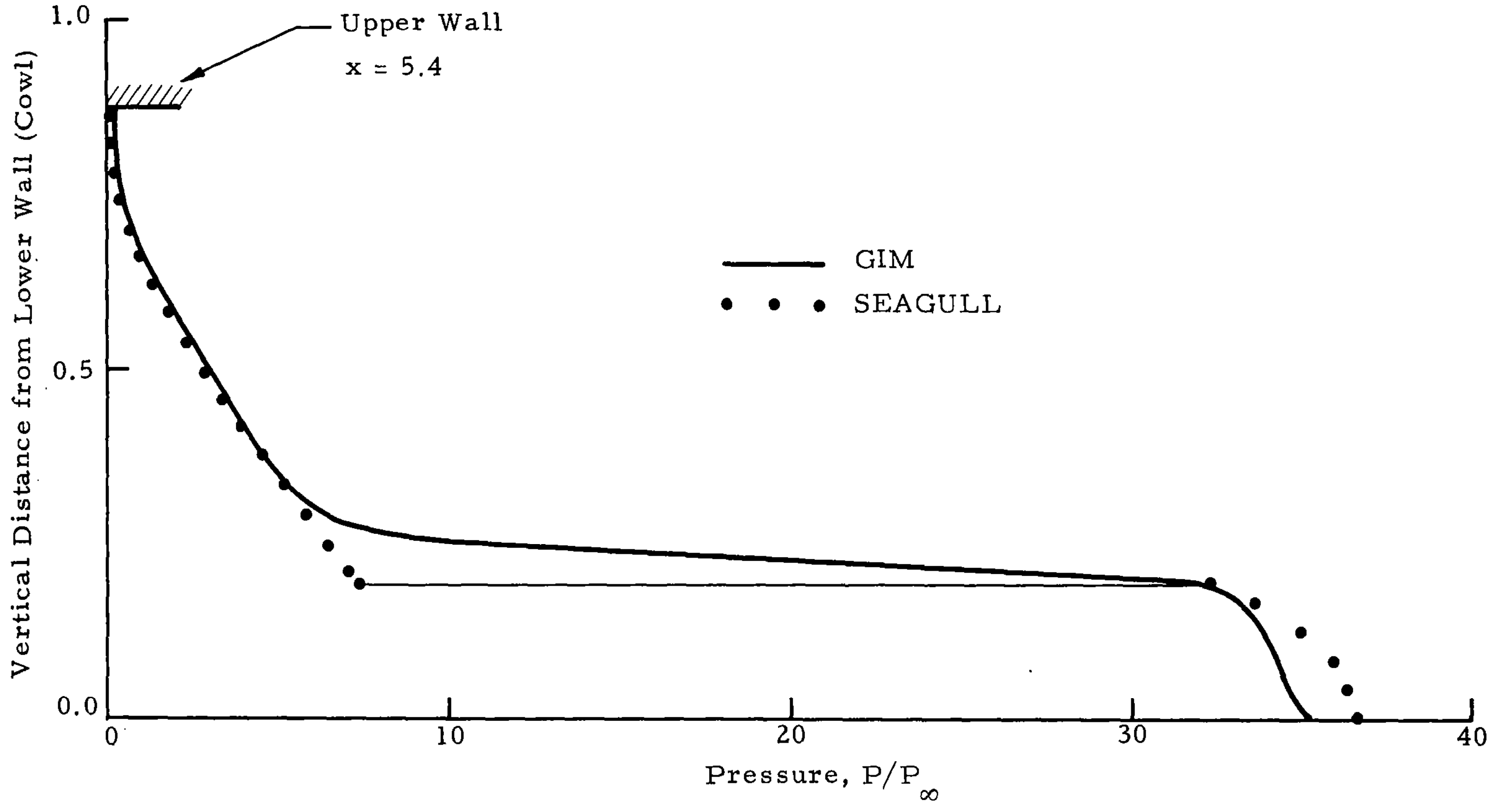


Fig. 2-22 - Pressure Variations Across the Nozzle.

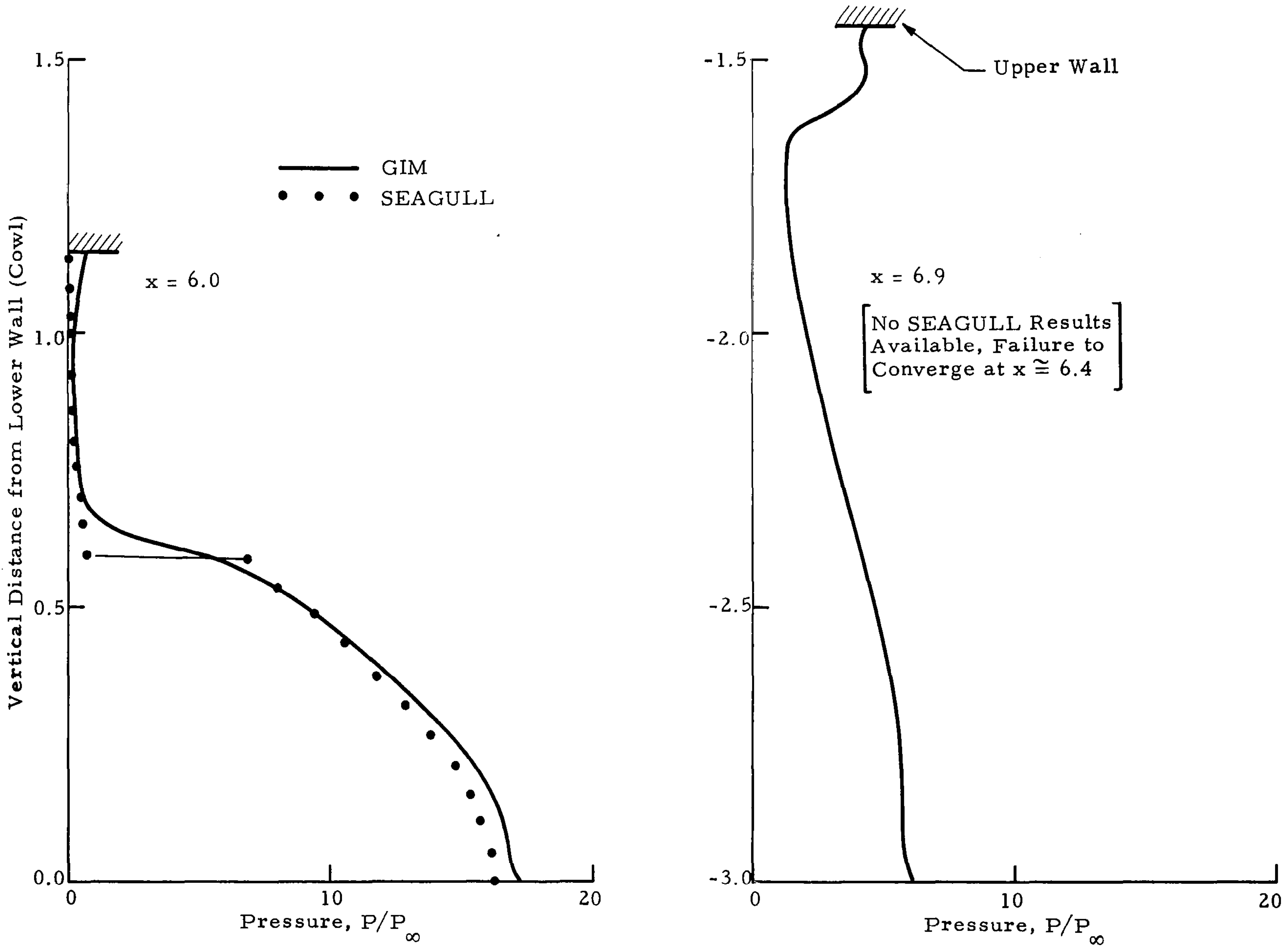


Fig. 2-23 - Pressure Variations Across the Nozzle.

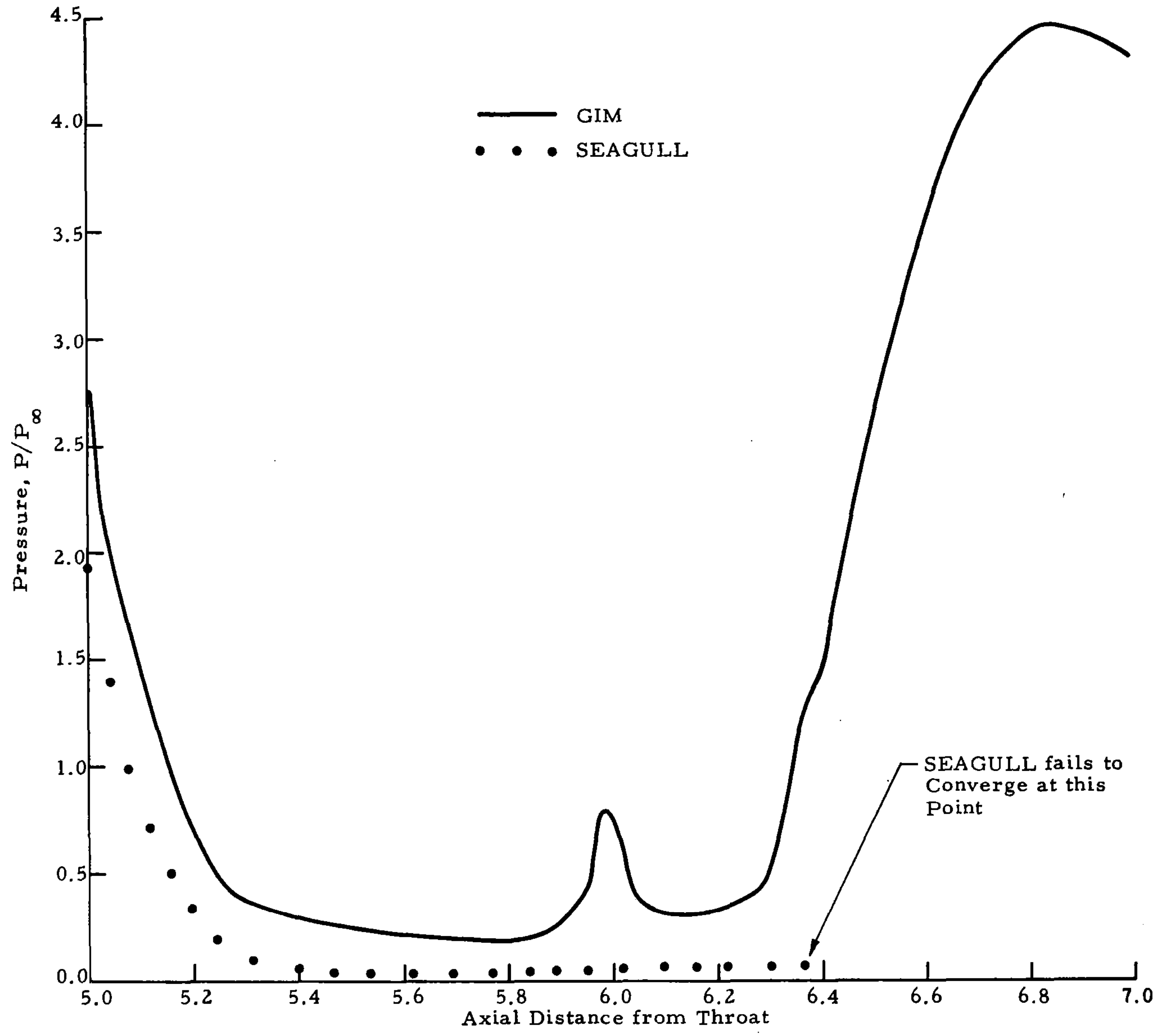


Fig. 2-24 - Pressure Variation on Upper Nozzle Wall.

3. INVESTIGATION OF LINEARIZED BLOCK IMPLICIT METHODS FOR THE GIM CODE

3.1 INTRODUCTION

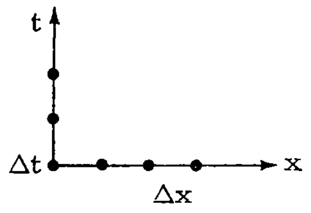
Numerical solution of the unsteady Navier-Stokes equations by explicit finite difference techniques has a number of disadvantages. The most serious one, from a practical engineering viewpoint, is the small time steps which are usually required to maintain stability. Computation of boundary layer flows at high Reynolds number requires fine grids near solid boundaries, hence very small time steps and long computer run times. One apparent cure for these difficulties is the use of implicit methods some of which are unconditionally stable for any size time step. These schemes are not without problems of their own in terms of their practical use. Among the major difficulties are the following:

1. Implicit finite differences, in general, lead to systems of nonlinear algebraic equations when applied to the Navier-Stokes equations. These must either be solved directly or linearized in some manner.
2. Direct linearization, via classical ADI processes, will destroy the Conservation Law Form of the Navier-Stokes equations and hence shock capture algorithms cannot be used.
3. Multi-dimensional implicit methods lead to very large systems of simultaneous algebraic equations. Even for linear systems, the efficient solution is not practical due to large size of the matrix coefficients.
4. Fully implicit methods cannot be programmed for efficient use on advanced vectorized machines such as the STAR, ILLIAC, or NASF.

Numerical treatment of the steady state parabolic form of the Navier-Stokes equations face many of the same difficulties as the elliptic form. The spatial marching step size is constrained by the small grid required to resolve boundary layers normal to a solid wall. Marching downstream great distances can result in impractically long run times. Implicit finite differences have the potential to eliminate the difficulties mentioned above.

3.2 ONE-DIMENSIONAL UNSTEADY DEVELOPMENT

The first item to be developed is the formulation of an implicit scheme which results in a linear algebraic system yet retains the conservation law form of the Navier-Stokes equations. This idea can be explored by considering the equations in one space variable, x, and the time coordinate, t.



Direct linearization is usually done by "lagging" certain of the nonlinear contributions by one time step. This destroys the conservation nature of the Navier-Stokes equations.

The case considered here is an elliptic boundary value problem in space and an initial value problem in time. The equations considered are:

Governing Equations

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} = \frac{\partial \tau}{\partial x}$$

$$U = \begin{pmatrix} \rho \\ \rho v \\ \rho \mathcal{E} \end{pmatrix} \qquad E = \begin{pmatrix} \rho v \\ \rho v^2 + P \\ (\rho \mathcal{E} + P)\, v \end{pmatrix} \tag{1}$$

$$\tau = \begin{pmatrix} 0 \\ \mu \dfrac{\partial v}{\partial x} \\ k \dfrac{\partial T}{\partial x} + \mu v \dfrac{\partial v}{\partial x} \end{pmatrix}$$

where

ρ	=	mass density	v	=	flow velocity
P	=	pressure	$\mathcal{E}$	=	total energy
μ	=	viscosity parameter $(2\,\mu^* + \lambda)$	k	=	thermal conductivity
t	=	time coordinate	x	=	space coordinate

$$P = (\gamma - 1)\,\rho\left[\mathcal{E} - v^2/2\right] \quad \text{ideal gas law}$$

<u>General Finite Difference Form</u>

This analysis will use the "delta" form of the flow variables

$$\begin{aligned} \Delta U^n &= U^{n+1} - U^n \\ \Delta E^n &= E^{n+1} - E^n \\ \Delta \tau^n &= \tau^{n+1} - \tau^n \end{aligned} \qquad (2)$$

where n is the time step index. All data are assumed known at $n = 0$. Solving for ΔU^n then allows the data at level n to be advanced to level $n+1$:

$$U^{n+1} = U^n + \Delta U^n$$

The class of finite-difference schemes considered can be written as follows:

$$\Delta U^n = \frac{\theta \Delta t}{1+\epsilon}\frac{\partial}{\partial t}(\Delta U^n) + \frac{\Delta t}{1+\epsilon}\frac{\partial}{\partial t}(U^n) + \frac{\epsilon}{1+\epsilon}\Delta U^{n-1} + O\left[(\theta - \tfrac{1}{2} - \epsilon)\,\Delta t^2 + \Delta t^3\right] \qquad (3)$$

The parameters θ, ϵ are used to generate a specific type of scheme. For $\theta = 0$, the scheme is fully explicit; $\theta > 0$ gives an implicit method. If $\epsilon = 0$, the scheme requires <u>two</u> data sets of storage at time levels $n, n+1$. If $\epsilon > 0$, then three levels are required to be stored, $n-1, n, n+1$.

If $\theta = 1/2 + \epsilon$, the scheme is second order accurate; and is first order otherwise. In this work, we are primarily concerned with

$$\theta = 1 \quad \epsilon = 1/2$$

which is a second order, implicit, three level scheme.

Development of the Scheme

The differential equation (1) is substituted into the general scheme (3) to get the following form:

$$\Delta U^n = \frac{\theta \Delta t}{1+\epsilon} \left[\frac{\partial}{\partial x} (-\Delta E^n + \Delta \tau^n) \right] + \frac{\Delta t}{1+\epsilon} \left[\frac{\partial}{\partial x} (-E^n + \tau^n) \right] + \frac{\epsilon}{1+\epsilon} \Delta U^{n-1} \tag{4}$$

By approximating the spatial derivatives, $\partial/\partial x$, by finite-differences, we get a set of nonlinear algebraic equations. The incremental flow variables, ΔE^n, $\Delta \tau^n$ are nonlinear functions (1) of the independent vector ΔU^n.

For our implicit scheme, this would require a simultaneous nonlinear algebraic equation solver. The best known methods are iterative ones which require long computer runs.

For this work, we will perform a linearization as follows to obtain a set of linear algebraic equations and use matrix methods for their solution. The main idea here is to linearize the algebraic equations, but retain the fully conservative nature.

Expanding E, τ in a Taylor series, we get

$$E^{n+1} = E^n + \left(\frac{\partial E}{\partial U}\right)^n (U^{n+1} - U^n) + O(\Delta t^2)$$

or

$$\Delta E^n = \left(\frac{\partial E}{\partial U}\right)^n \Delta U^n + O(\Delta t^2)$$

$$= A^n \Delta U^n + O(\Delta t^2)$$

and

$$\tau^{n+1} = \tau^n + \left(\frac{\partial \tau}{\partial U}\right)^n \Delta U^n + \left(\frac{\partial \tau}{\partial U_x}\right) \Delta U_x^n + O(\Delta t^2) \tag{5}$$

or

$$\Delta\tau^n = P^n \Delta U^n + R^n \Delta U_x^n + O(\Delta t^2)$$

where

$$U_x = \partial U/\partial x$$

The expression for $\Delta\tau$ can be rewritten in a more convenient form by expanding the x-derivative to get

$$\Delta\tau^n = (P - R_x) \Delta U^n + \frac{\partial}{\partial x} (R \Delta U)^n + O(\Delta t^2)$$

where

$$R_x = \partial R/\partial x$$

This form produces a linear system of equations with the same formal accuracy (Δt^2) as the nonlinear set. It does however, require evaluation of the Jacobians

$$A = \frac{\partial E}{\partial U} \qquad P = \frac{\partial \tau}{\partial U} \qquad R = \frac{\partial \tau}{\partial U_x} \tag{6}$$

and

$$\partial R/\partial x$$

Putting the Taylor series (5) into the scheme (4) gives the following expression:

$$\Delta U^n = \frac{\theta \Delta t}{1+\epsilon} \frac{\partial}{\partial x} \left[-A^n \, \Delta U^n + (P - R_x) \, \Delta U^n + \frac{\partial}{\partial x} (R \Delta U)^n \right] + \frac{\Delta t}{1+\epsilon} \frac{\partial}{\partial x} (-E^n + \tau^n) + \frac{\epsilon}{1+\epsilon} \Delta U^{n-1} \tag{7}$$

The last two terms on the right hand side of Eq. (7) are all explicit at time levels n, n-1. Denote this by D^n, and write Eq. (7) as follows:

$$\Delta U^n + \frac{\theta \Delta t}{1+\epsilon} \frac{\partial}{\partial x} \left[(A - P + R_x)^n \, \Delta U^n - \frac{\partial}{\partial x} (R \Delta U)^n \right] = D^n \tag{8}$$

For convenience, let

$$h = \frac{\theta \Delta t}{1+\epsilon} \qquad B = A - P + R_x$$

and write Eq. (8) as follows

$$\Delta U^n + h \left[\frac{\partial}{\partial x} (B^n \, \Delta U^n) - \frac{\partial^2}{\partial x^2} (R^n \, \Delta U^n) \right] = D^n \tag{9}$$

To see that the form Eq. (9) may be useful, we will now write it for node point "i" in space and use second order centered finite differences

$$\left. \frac{\partial f}{\partial x} \right|_i = \frac{f_{i+1} - f_{i-1}}{2 \Delta x} + O(\ x^2)$$
$$\left. \frac{\partial^2 f}{\partial x^2} \right|_i = \frac{f_{i+1} - 2\, f_i + f_{i-1}}{\Delta x^2} + O(\Delta x^2) \tag{10}$$

With these difference expressions, Eq. (9) can be written as follows:

$$\Delta U^n + h\left[\left(\frac{B^n_{i+1}\,\Delta U^n_{i+1} - B^n_{i-1}\,\Delta U^n_{i-1}}{2\Delta x}\right) - \left(\frac{R^n_{i+1}\,\Delta U^n_{i+1} - 2\,R^n_i\,\Delta U^n_i + R^n_{i-1}\,\Delta U^n_{i-1}}{\Delta x^2}\right)\right] = D^n_i \qquad (11)$$

Combining coefficients of each ΔU^n_i, ΔU^n_{i+1}, ΔU^n_{i-1} terms gives

$$\left(\frac{h}{2\Delta x}\,B^n_{i+1} - \frac{h}{\Delta x^2}\,R^n_{i+1}\right)\Delta U^n_{i+1} + \left(I + \frac{2h}{\Delta x^2}\,R^n_i\right)\Delta U^n_i + \left(-\frac{h}{2\Delta x}\,B^n_{i-1} - \frac{h}{\Delta x^2}\,R^n_{i-1}\right)\Delta U^n_{i-1} = D^n_i \qquad (12)$$

(where I is the 3x3 identity matrix)

Boundary values i = 1, and i = K must be treated separately due to the centered differences. For now we will let i = 2, 3, . . . k - 1 and worry about boundary conditions later.

The coefficients of the ΔU terms are 3x3 matrices which couple the three governing equations at each node point. There is an equation (12) for each node i = 2, 3, . . . k - 1.

To readily see the character of this system of linear algebraic equations, let

$$L^n_{i-1} = \frac{-h}{2\Delta x}\,B^n_{i-1} - \frac{h}{\Delta x^2}\,R^n_{i-1}$$

$$M^n_i = I + \frac{2h}{\Delta x^2}\,R^n_i \qquad (13)$$

$$N^n_{i+1} = \frac{h}{2\Delta x}\,B^n_{i+1} - \frac{h}{\Delta x^2}\,R^n_{i+1}$$

The linear algebraic system then has the form

$$L^n_{i-1}\,\Delta U^n_{i-1} + M^n_i\,\Delta U^n_i + N^n_{i+1}\,\Delta U^n_{i+1} = D^n_i$$

or in matrix notation:

$$\begin{bmatrix} M_2 & N_3 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ L_2 & M_3 & N_4 & 0 & 0 & & & 0 \\ 0 & L_3 & M_4 & N_5 & 0 & & & \cdot \\ \cdot & & & & & & & \cdot \\ \cdot & & & & & & & \cdot \\ \cdot & & & & & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & L_{K-3} & M_{K-2} & N_{K-1} \\ 0 & \cdot & \cdot & \cdot & 0 & L_{K-2} & M_{K-1} \end{bmatrix} * \begin{bmatrix} \Delta U^n_2 \\ \Delta U^n_3 \\ \Delta U^n_4 \\ \cdot \\ \cdot \\ \cdot \\ \Delta U^n_{K-2} \\ \Delta U^n_{K-1} \end{bmatrix} = \begin{bmatrix} D^n_2 \\ D^n_3 \\ D^n_4 \\ \cdot \\ \cdot \\ \cdot \\ D^n_{K-2} \\ D^n_{K-1} \end{bmatrix} \qquad (14)$$

The system (14) will be termed "block tridiagonal." The individual matrices are full 3 x 3 arrays but they are arranged in a tridiagonal manner in the full matrix. The block arrangement occurs due to the linearization scheme used. This effectively couples the three differential equations at each node point. The boundary values for $i = 1$, and $i = K$ have not been treated. This is an additional development item.

The advantages of a system like Eq. (14) are:

1. The Conservation Law form has been retained.
2. Block tridiagonal systems are not much more costly to solve than pure tridiagonal systems.
3. Operations like (14) can be vectorized for use on STAR-like machines.

Formulation of this scheme requires the analytical evaluation of the Jacobian matrices A, P, R. A brief look at these operations now follows.

Calculation of the Matrices

The final matrices needed are L, M, N in Eq. (13). These are made up of combinations of B, R matrices from Eq. (5).

$$R^n = \left(\frac{\partial \tau}{\partial U_x}\right)^n$$

$$B^n = A^n - P^n + R_x^n \tag{15}$$

$$A^n = \left(\frac{\partial E}{\partial U}\right)^n \qquad P^n = \left(\frac{\partial \tau}{\partial U}\right)^n$$

$$R_x^n = \left(\frac{\partial R}{\partial x}\right)^n$$

The matrices A, R will have relatively simple elements (as we shall see), but the P, R_x matrices will be quite complex. For now we will assume that the viscous coefficients are <u>constants</u>; hence we will see that

$$P - R_x = 0$$

(See Beam-Warming paper, Ref. 3).

We then need to analytically evaluate A, R, where

$$A_{ij} = \frac{\partial E_i}{\partial U_j} \qquad R_{ij} = \frac{\partial \tau_i}{\partial U_{x_j}}$$

The algebra for these operations is straightforward and is not included here. The final results are:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{(\gamma-3)}{2} v^2 & (3-\gamma) v & \gamma-1 \\ -\gamma v \mathcal{E} + (\gamma-1) v^3 & \gamma \mathcal{E} + \frac{3}{2}(1-\gamma) v^2 & \gamma v \end{bmatrix}$$

$$R = \frac{1}{\rho} \begin{bmatrix} 0 & 0 & 0 \\ -\mu v & \mu & 0 \\ -(\mu - \frac{k}{C_v}) v^2 - \frac{k}{C_v} & (\mu - \frac{k}{C_v}) v & \frac{k}{C_v} \end{bmatrix} \tag{16}$$

where C_v is the (constant) specific heat at constant volume, γ is the ratio of specific heats and k is the thermal conductivity.

Summary of Computational Procedure

1. Set initial data at $t = 0$ for all nodes $i = 1, 2, \ldots K$.
2. Form the vectors U, E, τ.
3. Compute D by explicit differences (Eq. (8)).
4. Evaluate A, R matrices from Eq. (16).
5. Form the L, M, N matrices from A and R (Eq. (13)).
6. Modify for boundary values.
7. Call TRIDAG in the GIM code logic to solve the block tirdiagonal system for ΔU^n.
8. Advance solution vector to (n+1)

$$U^{n+1} = U^n + \Delta U^n$$

9. Repeat the process to step 2 for a specified number of steps or until $|\Delta U^n| < \delta$ for convergence to steady state.

3.3 ONE-DIMENSIONAL LINEARIZED BLOCK IMPLICIT RESULTS

The procedure outlined in Sections 3.1 and 3.2 was subsequently coded and checked out. The equations were modified slightly to handle the problem of an expanding duct, quasi one-dimensional, by the inclusion of the area terms. This permits the computation of flows other than just the trivial case of constant property flow through a constant area duct. Three cases were considered in order to check out and prove the method. Consider Fig. 3-1 where the simplest case is when the inflow conditions are fixed at the upstream end of the duct. For completely subsonic flow, elementary considerations indicate that the outflow at the downstream end of the duct has a unique solution. Consider for the moment that the flow is controlled entirely by the inflow conditions and the out-flow conditions are permitted to develop freely. Of course, it is known that physically one could change the back pressure at the downstream end and this would affect conditions at the upstream end. But, computationally, we specify the inflow conditions and therefore all the flow properties are uniquely determined. The same reasoning applies to the case where the flow is completely supersonic. In this case there exists the choking effect which means that when the back pressure is lowered below the limiting value no upstream effect is felt. If however, the back pressure is raised, the situation develops where a normal shock moves into the duct with its positioning depending upon the back pressure. Thus for fixed inflow conditions for the supersonic case an unique solution depends upon the outflow pressure. Since so much is known analytically about this quasi one-dimensional case it was deemed a reasonable test with which to evaluate the linearized block implicit (LBI) scheme.

In order that the LBI scheme could be applied to all three cases, i.e., including the strong shock case, pseudo viscous effects were included in the original coding in terms of numerical diffusion cancellation (NDC) terms. The first case computed was for fully supersonic flow through the expanding duct. The LBI scheme worked well reproducing the analytical results within about 2% over the length of the duct. The case was initially run at a Courant number of one. Subsequent runs were made at larger Courant numbers up to 3

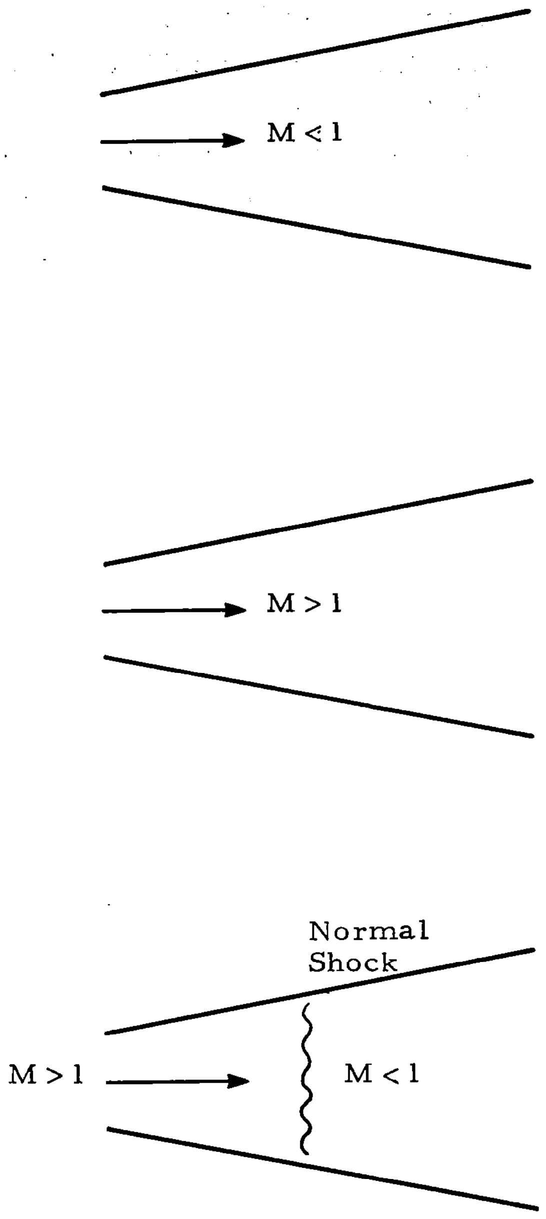


Fig. 3-1 - One Dimensional Duct Configuration

with good results. Increasing the CFL multiplier further caused rapid deterioration of the solution and ultimate destruction of the case (it blows up).

Theoretically the implicit solution should work for very large Courant numbers. Mathematically this is, of course, true but it ignores the physics of the situation. To verify that the solution was correct the complete derivation was double checked, the coding was rechecked and nothing was found wrong. At first it was thought that the non-dominance of the main diagonal might be causing matrix ill-conditioning. The super- and sub-diagonals are both proportional to the step size while the main diagonal remains constant (at least for the inviscid equations). A natural conclusion might then be drawn that, as the step size is increased, non-dominance could occur such that the solution of the block matrices loses accuracy thus destroying the solution.

To test out this theory some numerical experiments were carried out. First, an unblocked scalar matrix with three diagonals was used. A known solution was fed into the matrix reduction scheme and the non-dominance factors between the main and other diagonals were increased gradually. The case was run on the PDP-11, single precision arithmetic and inaccuracies did show up in the sixth place for even a 2 to 1 non-dominance ratio. At 10^6 to 1, inaccuracies occur in the first and second places and at 10^9 to 1, order of magnitude inaccuracies were produced. Using double precision arithmetic on the PDP-11 or running the case on a CDC 7600 produced no inaccuracies whatsoever. Thus it is concluded that scalar matrices manipulated on high precision computing equipment have no accuracy problems associated with diagonal non-diminance.

The same type of numerical experiments were then conducted with the block matrices. The CDC 7600 was used in order to eliminate any inaccuracy due to less precise computing equipment. Non-dominance ratios on the order of 10^6 to 1 were necessary to generate errors in the fifth and sixth place. Since the suspected non-dominance caused by increasing the step size would only be of order 10, it is concluded that the reason the case would not run at large Courant numbers is due to the problem physics and is not related to the mathematics of diagonal non-dominance.

Subsequent consultations with NASA-Ames personnel (Robert Warming and Richard Beam) indicated that they saw no accuracy problems related to non-dominant diagonals and they believe the problem with using large Courant numbers is due to physically unrealistic propagation of pressure signals which then cause oscillatory behavior and eventually a negative pressure. Two different solvers were used to eliminate the possibility of an error in the coding. The two solvers, one from Lockheed-Huntsville and one from Ames Research Center, produced identical results. A fourth order damping term was appended to the RHS to help alleviate some of the oscillatory behavior.

Ames indicated that in all their calculations with centered differences, fourth order damping was used. A fourth derivative term was therefore approximated and added to the RHS of the equations. The numerical diffusion cancellation terms were then dropped, except for the cases with shock. Use of the damping term eliminated some of the spatial oscillation but is highly dependent upon the value of an arbitrary coefficient which can vary between 0 and 2. If too small a value is used the parameters oscillate, if too large a value is used the solution is overdamped and becomes linearized. A compromise value used throughout this study was 0.1 which worked quite well for most of the cases analyzed.

Another idea that was investigated involved the use of a MacCormack operator to compute the RHS. It is well known that the two-step MacCormack operator gives second order accuracy and is very stable. This scheme worked quite well and eliminated the necessity of including the fourth order damping term.

To this point all the calculations were done using three point centered difference approximations to the derivatives. This results in the basic block tridiagonal scheme. One can also formulate the equation set based upon a backward difference approximation which then results in a block bidiagonal scheme. This approach worked very well and, as is well known, has excellent stability characteristics. Its major limitation is that it is only first order accurate and generally is applicable only to supersonic flows. This scheme is inherently stable for any step size, and several cases were run at Courant numbers of 1000.

Instead of using a pure centered scheme which has stability problems or a backward difference approximation that is only first order accurate, a combination of weighted differences was evaluated. Several combinations of weight factors were investigated, such as 2/3 centered plus 1/3 backward, and generally it was found that a slight increase in the Courant number could be obtained over that required for the pure centered scheme. Accuracy remained about the same as the centered differencing scheme.

As Fig. 3-2 shows, the solution technique previously discussed produces very reasonable results including the location of the normal shock in the diverging duct.

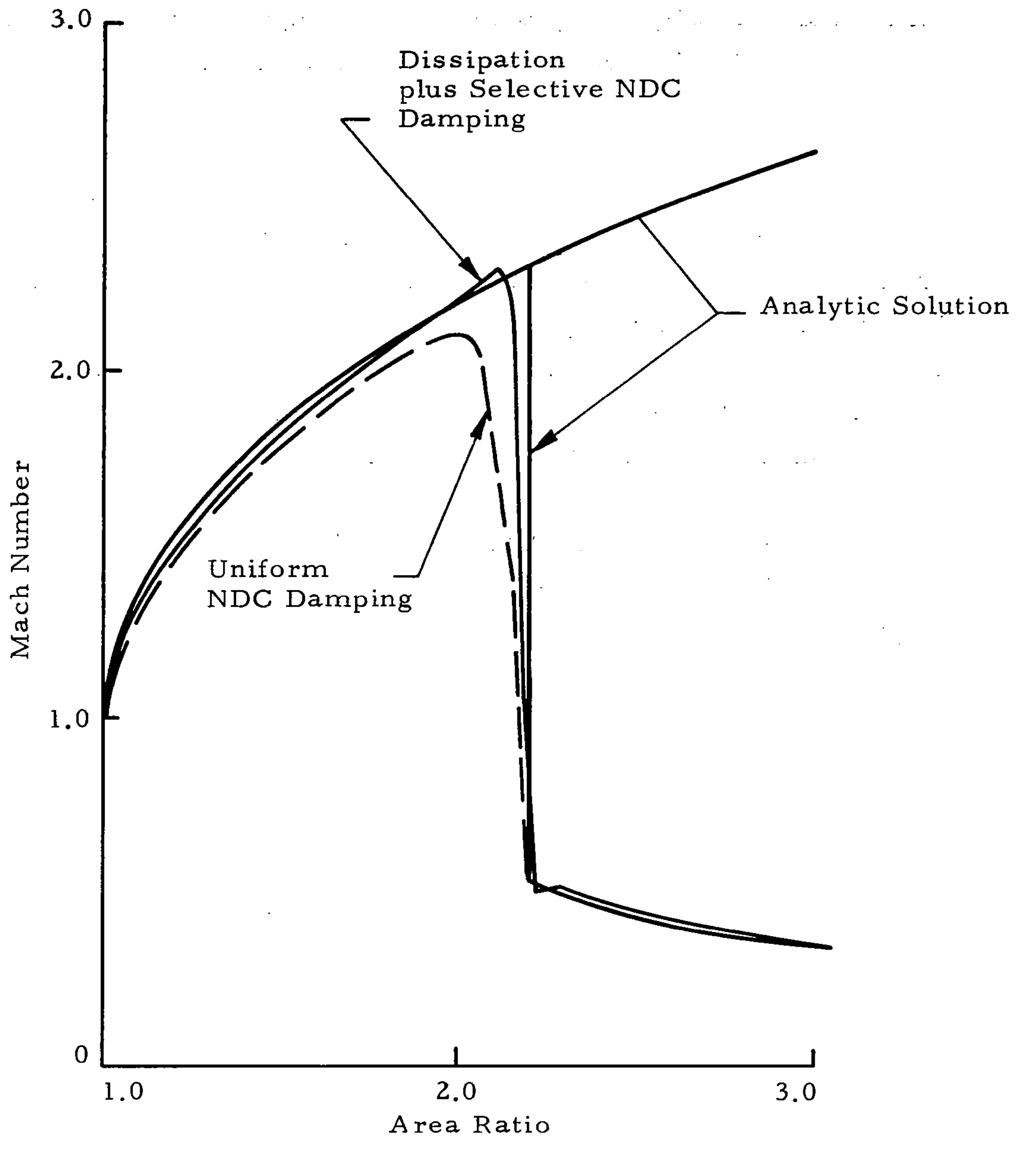


Fig. 3-2 - Quasi-One Dimensional Results.

4. DEVELOPMENT OF GIM/STAR SPATIAL MARCHING ALGORITHMS

4.1 INTRODUCTION

The GIM/STAR SE-1 code treats the full elliptic flow field using explicit finite difference methods. This technique is applicable to a large range of fluid dynamics problems and has been successful in computing a number of these. The current code can be an "overkill" for some problems of interest in that a full elliptic treatment is not necessarily required. A parabolized, spatial marching algorithm could provide accurate flow fields for these situations and would be considerably more economical.

The elliptic code is constrained by two items which restrict its use on large three-dimensional viscous flows:

1. The time step in explicit schemes is restricted by the CFL and viscous stability limit. This is usually controlled by the small grid sizes normal to no-slip boundaries. If inviscid, free slip boundaries can be used, i.e., ignore the boundary layer, then the severity of this constraint decreases. The implicit, linearized block methods described in Section 3 provide a possible remedy for the time step difficulty in the elliptic code.

2. The large amount of data storage needed for three-dimensional viscous flows causes large "page faulting" on the STAR machine. Any finite difference method, explicit or implicit, still requires the large data base. A GIM/STAR code with a parabolic spatial marching algorithm would not attack as many kinds of problems as the elliptic version but would allow large three-dimensional viscous flows to be treated with no page faults on STAR.

The intent of this research is to provide both an elliptic, time-dependent GIM/STAR code and a hyperbolic/parabolic spatial marching version. It is not too difficult to conceive of the future codes which could contain switching logic to automatically change from elliptic to parabolic etc., depending on the physics of the flow, however this will not be attempted here. The section will review the available classical parabolized methods and their problems, and then present an idea believed to be new for computing quasi-parabolic flows.

4.2 CLASSICAL PARABOLIC APPROACHES

Most of the literature on spatially marching schemes, hyperbolic or parabolic, treat equations which have been transformed to a Cartesian computation grid which is uniform. The space marching can then be done in much the same manner as time marching. This is a good approach if a single transformation exists for the full flow domain. The GIM code strategy has been to compute in the physical domain whereby completely arbitrary geometries can be treated. This approach presents a problem in developing a space marching algorithm, i.e., the fact that the geometry changes in the marching coordinate direction. This is akin to a GIM unsteady time marching scheme whereby the geometry is allowed to change with time. If we are to keep the GIM strategy of arbitrary geometries, then a space marching scheme must be developed which will account for the geometric variations in the streamwise direction, i.e., non-uniform computational domain.

The recent work of Roberts and Forester (Ref. 5) use a boundary-fitted computational mesh in a parabolic code for ducts of arbitrary cross section. Their algorithm for solving the equations appears to be a refinement of the classical method of Patankar and Spalding (Ref. 6). Rubin and Lin (Ref. 7) presented a nonlinear, iterative finite difference method for three-dimensional viscous flows. A parabolic method using a block implicit type scheme was given by Hirsh (Ref. 8). The solutions were restricted to supersonic flow (shear layers) of the free mixing type. Lubard and Helliwell (Ref. 9) calculated flows on cone at angle of attack using a parabolized method. This paper discussed some of the inherent difficulties with singularities, ambiguities and departure solutions which arise in parabolized algorithm. The paper discusses explicit and implicit schemes for parabolic marching flows.

Lin and Rubin (Ref. 10) presented a method using psuedo-time relaxation with a space-centered implicit differencing technique. They discuss many of the problems inherent in "pure" parabolic marching and show how time relaxation can eliminate departure solutions. The GIM technique, although developed independently of Lin and Rubin, also employs time relaxation but with an explicit, one-sided, predictor-corrector scheme and arbitrary three-dimensional geometries. The second order backward-forward, backward-backward explicit scheme of the GIM code is also a unique approach to parabolic marching solutions.

For problems in which viscous terms can be neglected entirely and the main flow direction remains supersonic, we would like the capability in the GIM code to resort to a simple hyperbolic algorithm. The classical methods presented in the literature for parabolic and hyperbolic flows are drastically different because of the treatment of the pressure terms in the marching direction. As long as the flow is inviscid and supersonic, the axial pressure terms can be treated exactly. However, for subsonic flow, for example, several problems arise in applying a hyperbolic algorithm to the parabolic equations. This is, however, the approach that would be most general, if the "parabolic pressure" problem can be treated.

Certain assumptions must be made in using a spatial marching technique.

- There must exist a dominant flow direction in which to march. There can be no flow back upstream, i.e., no recirculation in the streamwise direction.
- Stress terms are not allowed to act on the cross planes: i.e., there can be no second order terms (diffusion, viscosity) in the marching coordinate.
- The downstream pressure field must not be allowed to propagate upstream.

There are a number of strong implications in these assumptions. A supersonic, inviscid flow satisfies them all. A supersonic viscous flow will conform to the assumptions if the viscous terms are dropped in the marching coordinate direction.

Consider now the problem of spatial marching in a subsonic viscous flow. The first two of the assumptions can be met by simply not allowing any flow reversal problems to be attempted and dropping all streamwise diffusion terms. The downstream pressure field can still feed back through a subsonic stream. One obvious approach is to drop the streamwise pressure gradient term. This would satisfy the third assumption, but it appears a serious matter to simply drop this important term.

Another approach commonly used is to provide a separate, explicit equation for the pressure and use windward, one-sided differences. The most exact way is to compute the conserved flux parameters and then "decode" for the velocity, density and energy and compute the pressure from a state relation. The ideal gas law, a set of equilibrium thermodynamic relations or Boussinesq equations, is used to couple the state variables. Each of these approaches contains inherent difficulties which render their general use questionable. The following is a summary of some of these problem areas with classical parabolic schemes.

Zero Axial Pressure Gradient

This does not cause any significant numerical problems in computing a flow field. It does however create a major problem in that the computed answers are probably wrong for most flow fields. A mixed supersonic/subsonic flow, for example, with a shock wave crossing the flow field cannot be computed at all because of the large axial (and radial) gradients. Some researchers still proceed to use this approach and try to justify it.

Exact Pressure Treatment

The rigorous way to compute the parabolized equations is to include the pressure in the conservation variable state vector for the momentum equations. A state equation can then be used to "decode" for the pressure. The advantages of the approach are that: (1) fully conservative differencing can be used; (2) shock capture algorithms are applicable; and (3) an auxiliary differential equation for the pressure is not needed. However, there are major problems with the "exact" treatment of the parabolic pressure.

- One-sided upstream differences must be used
- The "decode" is ambiguous at Mach = 1 since two roots appear for the velocity (or pressure)
- Real viscous, no-slip walls cannot be treated since the decode is singular.
- Flows with a quiescent part, such as jets exhausting into an ambient, motionless atmosphere cannot be treated because of the singularity for zero velocity.

Consider the two-dimensional parabolic system

$$\frac{\partial E}{\partial x} + \frac{\partial F(E)}{\partial y} = 0$$

where x is the marching coordinate, y the cross plane (or radial) coordinate, E is the state vector of conservation variables and F is a nonlinear (viscous plus convective terms) function of E.

A typical state vector E for the parabolized Navier-Stokes equations is

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho u v \\ (\rho\mathcal{E} + P) u \end{bmatrix}$$

$$P = (\gamma - 1)\rho \left[\mathcal{E} - \frac{u^2 + v^2}{2}\right]$$

Here, u is the axial velocity, v is the cross plane coordinate velocity, ρ is density, $\mathcal{E}$ is the total energy per unit volume and P is the pressure. Suppose a calculated value for the E vector exist at a plane $X = X_i$. It is now required to "decode" for the primitive variables. The following is one decode procedure that can be used in computer codes.

(1) $v = E_3/E_1$

(2) $u = \frac{\gamma}{\gamma+1}\left[\left(\frac{E_2}{E_1}\right) \pm \sqrt{\left(\frac{E_2}{E_1}\right)^2 - 2\left(\frac{\gamma^2-1}{\gamma^2}\right)\left(\frac{E_4}{E_1} - \frac{E_3^2}{2E_1^2}\right)}\right]$

(3) $\rho = E_1/u$

(4) $\mathcal{E} = \frac{E_4}{\gamma E_1} + \frac{(\gamma - 1)}{2\gamma}\,(u^2 + v^2)$

(5) $P = (\gamma = 1)\,\rho\left[\mathcal{E} - \frac{u^2 + v^2}{2}\right]$

Two problems are immediately obvious:

- The radical in the u velocity decode causes an ambiguity. It can be easily shown that the correct decode is to take the + sign for u supersonic and the - sign if u is subsonic. In mixed flows, the sonic nature of a grid point is not known a priori. This Mach = 1 ambiguity prohibits a general parabolic marcher from being developed using the classical notions. See Section 5 of Appendix B.
- The axial velocity, u, cannot be zero or the decode is singular. The axial component must be zero, however, if a real wall is to be put into the problem. All classical parabolic codes simply use some wall functions or resort to inviscid slip conditions to avoid the singularity.

A third difficulty, which is not so obvious, is that attempts to use implicit methods to march the solution downstream often fail. The reason is that the boundary conditions are not treated exactly, and these errors build up as the streamwise coordinate is traversed. Often, the explicit differencing of points near the wall is used as a patchwork way of circumventing the boundary condition difficulty.

Explicit Treatment of Pressure

This is the most widely used of the parabolic procedures and its origination is usually attributed to D. B. Spalding. The idea is to provide an explicit differential equation for the pressure field in addition to the basic conservation laws. This is usually a Poisson-type relation obtained from combining continuity and momentum equations. Satisfaction of local mass conservation is generally the criteria used for convergence of the elliptic Poisson equation.

In general, a state vector will have the following appearance:

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho u V \\ (\rho \mathcal{E} + P) u \\ P \end{bmatrix}$$

where the E_5 component now represents the differential equation solution for pressure from whatever means.

Now note the difference in the "decode" from the exact treatment case:

$$\begin{aligned} P &= E_5 \\ u &= \frac{E_2 - E_5}{E_1} \\ v &= E_3/E_1 \\ \rho &= \frac{E_1^2}{E_2 - E_5} \\ \mathcal{E} &= \frac{E_4}{E_1} - P/\rho \end{aligned}$$

The Mach = 1 ambiguity is no longer present as the radical does not appear. Thus mixed subsonic/supersonic flows can be computed without a priori knowledge of the Mach number. Note however, that the decode still contains the axial velocity in the denominator. Real solid walls cannot enter if viscous boundary conditions are used.

This "explicit pressure" treatment requires solution, at each plane, of a Poisson-type equation. Thus, an iteration between planes is required before moving on down to the next plane. Even with its inherent bad points, this approach remains the most successful and widely used parabolic algorithm.

4.3 THE QUASI-PARABOLIC IDEA

The results of the initial investigation of a parabolic/hyperbolic GIM code led to the conclusion that there just is not a good approach being used today that fits the GIM code strategy. Three basic requirements were placed on a GIM/parabolic algorithm:

- The geometric treatment must be applicable to arbitrary shapes.
- The same basic algorithm should be applied to both hyperbolic and parabolic flows and be capable of eventual coupling with an automated algorithm for switching back and forth to the elliptic solver.
- The algorithm should be readily vectorizable to realize the speed gain from using the STAR computer.

In terms of a "classical" parabolized spatial marching algorithm, several geometric approaches were investigated.

The first approach considered would generate the geometry plane by plane as the solution evolves, assembling the elements locally at each step. This would of course mean that the GEOMETRY module would be called at each integration step, thus coupling the geometry and the flow. An advantage of this approach is that only the amount of geometry needed would be computed and stored at any cycle of the calculation. This would reduce the computer storage and reduce the large input/output problems. However, this approach would also require considerable reprogramming to make the GEOMETRY module of GIM a subprogram to the INTEGRATION module.

A second approach appears to be a treatment of the geometry uncoupled from the flow. This means that all geometry, transformations and element assembly would be done before the flow field is integrated. The matrix data would be read from a stored file for each cross plane as needed. The advantages of this approach are the geometry is computed only once for a given configuration, the geometry module can be separate (as it is now), from the integration module and the grid could be inspected for desirable character prior to computing an expensive flow field. Disadvantages are that the basic character of the flow must be analyzed a priori to place grid planes in desirable locations, and data must be read from files at each cross plane which could effect the thru-put time on the computer.

A third possiblity is to switch to computing in a transformed computational space. This makes the marching algorithm straightforward but forfeits one of the major advantages of GIM — completely arbitrary geometries.

Approach 2 was selected as the best compromise and also provides the ultimate capability of elliptic-parabolic switching discussed earlier.

The classic algorithms for treating the parabolic pressure field were deemed unsatisfactory. The following idea evolved from this research. The approach is termed "Quasi-Parabolic" and arose from the requirement of eliminating the ambiguities and singularities of existing methods.

The basic idea is to combine the classical parabolic marching approach with a "quasi time" relaxation. The parabolic-march procedure greatly reduces the amount of computer storage compared to a fully elliptic field. The time relaxation form of the equations eliminates the "decode" ambiguity associated with the parabolic pressure problem and allows velocity boundary conditions at solid walls to be treated. The equations used in the QP method are the time-averaged full Navier-Stokes, but with all second order terms dropped in a quasi-marching coordinate. Another way to view the QP equations is to take the parabolized Navier-Stokes and add back "psuedo time" derivatives. The QP solution procedure, as any parabolic marcher, thus allows no downstream diffusion effects or pressure wave feedback through a subsonic flow. The solution is assumed known at upstream data planes, 1, 2, ...N-1, and the solution is sought at plane N with no knowledge of plane N+1. "Psuedo Time" relaxation, is used to obtain the solution at plane N in terms of the (converged) solution at a number of upstream data planes. Backward differences of some type, (second order) must be used to prohibit downstream feedback. So the QP algorithm is not a classical space marching scheme, and is also not a time-dependent elliptic method. It is somewhat of a hybrid technique which combines the better features of two approaches and eliminates the bad ones.

The GIM/STAR elliptic code will converge a case in 500 to 1000 steps if the initial guess is chosen reasonably close to the answer. Also, GIM/STAR is relatively cheap to run, if the problem size is small enough to fit into memory and not require large page faults. The QP algorithm relieves both of these difficulties to some extent. By storing only a small number of data planes (and not the entire elliptic field) the large page fault problem is gone. The QP marching procedure can also assign a reasonable guess to the N^{th} data plane since it knows the upstream converged solution, i.e., guess it is equal to the $N-1^{st}$ plane or extrapolated in some way. This should allow the time relaxation to converge very rapidly. If an implicit time-relaxer is used with the QP algorithm (with steps many times the CFL), the relaxation should go even faster.

The QP method allows an exact treatment of the Parabolic pressure field. No ambiguity exist in the QP decode at Mach = 1 (since it is "quasi-time" dependent) and no-slip walls can be treated exactly, i.e., boundary layers. The QP algorithm eliminates many of the bad features of pure parabolic methods.

One obvious disadvantage of the QP approach is the planewise iteration (time relaxation) which must be done. This can be time consuming on the machine, and a good criterion for convergence must be used to avoid error propagation downstream. Spalding's method suffers from this same planewise iteration to correct the pressure as well; other linearization schemes such as Roberts and Forester (Ref. 5) which use the conservative equations also suffer. Planewise iteration is not uncommon in most parabolic methods, thus the QP scheme is no better or worse in this respect. A linearized block implicit scheme, as discussed in Section 3, appears to be very attractive for performing the quasi-time relaxation.

Figure 4-1 shows the QP form of the three-dimensional Navier-Stokes equations in Cartesian coordinates. Note that these are the classical parabolized form plus a psuedo-time derivative. Included are global mass conservation, three components of momentum conservation, total energy and an equation for conservation of individual species in a binary mixture. Figure 4-2 is a typical computation molecule for a QP type marching. Assume that all flow variables are known at planes 1, 2...K and the solution is sought at plane K + 1. If backward differences in x are used, (first order, second order, etc.), then the scheme of Fig. 4-3 allows no downstream feedback, and allows plane K + 1 to be uniquely determined from upstream information, i.e., quasi-parabolically.

Now consider the ultimate, not immediate, implications of such an algorithm. A flow field could be marched out quasi-parabolically from an initial data plane 1 to plane K, where K is set a priori by the users. An imbedded elliptic region is encountered between planes K and K + M. The number of planes that the QP algorithm can treat on any given sweep is not

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0$$

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho\mathcal{E} \\ \rho C \end{bmatrix}$$

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ (\rho\mathcal{E} + P)u \\ \rho uC \end{bmatrix}$$

$$F = \begin{bmatrix} \rho v \\ \rho vu - \tau_y \\ \rho v^2 + P - \tau_{yy} \\ \rho vw - \tau_{yz} \\ (\rho\mathcal{E} + P)v - u\tau_y - v\tau_{yy} - w\tau_{yz} - q_y \\ \rho vC - R_y \end{bmatrix}$$

$$G = \begin{bmatrix} \rho w \\ \rho wu - \tau_z \\ \rho wv - \tau_{yz} \\ \rho w^2 + P - \tau_{zz} \\ (\rho\mathcal{E} + P)w - u\tau_z - v\tau_{yz} - w\tau_{zz} - q_z \\ \rho wC - R_z \end{bmatrix}$$

$$\tau_{yy} = 2\mu \frac{\partial v}{\partial y} + \lambda \left(\frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right)$$

$$\tau_{zz} = 2\mu \frac{\partial w}{\partial z} + \lambda \left(\frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right)$$

$$\tau_y = \mu \left(\frac{\partial u}{\partial y} \right)$$

$$\tau_z = \mu \left(\frac{\partial u}{\partial z} \right)$$

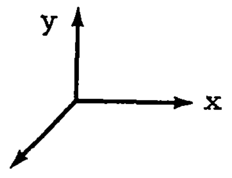
$$\tau_{yz} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right)$$

$$q_y = k \frac{\partial T}{\partial y} + \rho\mathcal{D} (h_1 - h_2) \frac{\partial C}{\partial y}$$

$$q_z = k \frac{\partial T}{\partial z} + \rho\mathcal{D} (h_1 - h_2) \frac{\partial C}{\partial z}$$

$$R_y = \mathcal{D} \frac{\partial C}{\partial y}$$

$$R_z = \mathcal{D} \frac{\partial C}{\partial z}$$



(x is QP coordinate)

Fig. 4-1 - Three-Dimensional Quasi-Parabolic Navier-Stokes System in Cartesian Conservation Law Form.

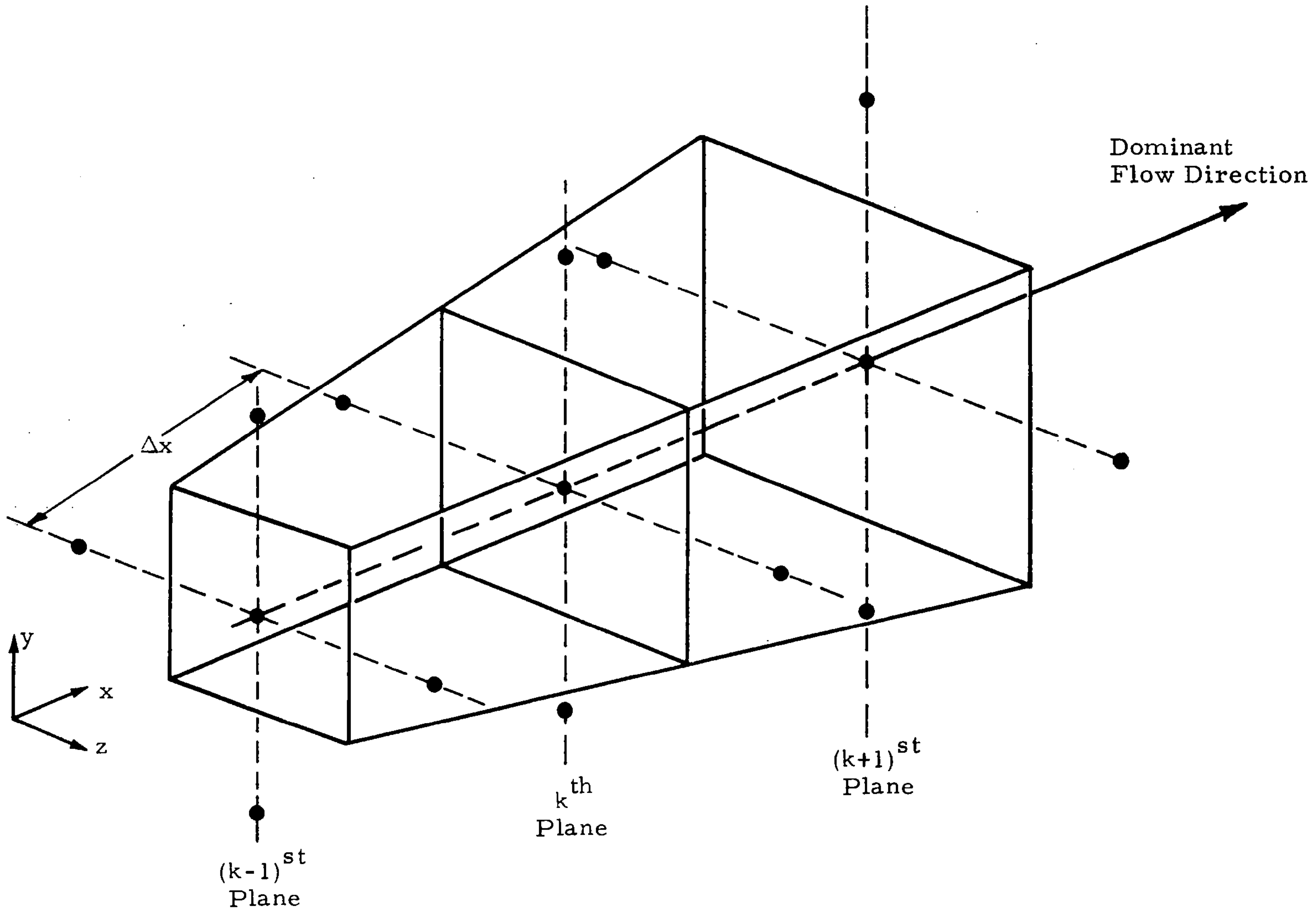


Fig. 4-2 - Symbolic GIM Code Computation Molecule.

<u>Equation</u>

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0$$

<u>Predictor</u>

$$\hat{U}^{n+1}_{i,j,k} = U^{n}_{i,j,k} - \frac{3}{2}\frac{\Delta t}{\Delta x}\left[E^{n}_{i,j,k} - E^{*}_{i,j,k-1}\right] + \frac{1}{2}\frac{\Delta t}{\Delta x}\left[E^{*}_{i,j,k-1} - E^{*}_{i,j,k-2}\right]$$

$$- \frac{\Delta t}{\Delta y}\left[F^{n}_{i+1,j,k} - F^{n}_{i,j,k}\right] - \frac{\Delta t}{\Delta z}\left[G^{n}_{i,j+1,k} - G^{n}_{i,j,k}\right]$$

<u>Corrector</u>

$$U^{n+1}_{i,j,k} = \frac{1}{2}\left\{U^{n}_{i,j,k} + \hat{U}^{n+1}_{i,j,k} - \frac{3}{2}\frac{\Delta t}{\Delta x}\left[\hat{E}^{n}_{i,j,k} - E^{*}_{i,j,k-1}\right] + \frac{1}{2}\frac{\Delta t}{\Delta x}\left[E^{*}_{i,j,k-1} - E^{*}_{i,k,k-2}\right]\right.$$

$$\left. - \frac{\Delta t}{\Delta y}\left[\hat{F}^{n}_{i,j,k} - \hat{F}^{n}_{i-1,j,k}\right] - \frac{\Delta t}{\Delta z}\left[\hat{G}^{n}_{i,j,k} - \hat{G}^{n}_{i,j-1,k}\right]\right\}$$

Fig. 4-3 - The Quasi-Parabolic Scheme Finite Difference Equations.

restricted to <u>one</u>. Simply specify single plane marching up to plane K, switch to the elliptic operator on the next M planes, and then parabolically march from the (K+M)th plane to the final Nth plane. The switching can then be done "automatically," but the user must still determine the location to perform the switching. Eventually, perhaps, an algorithm could be written to detect the onset of a separation bubble, flow reversal, or other elliptic phenomena. This is not being considered at this time, but only the fact that the capability is within the framework of the QP algorithm.

Advantages of the QP Algorithm Outlined

- There is no special treatment required of the parabolic pressure field. It is handled exactly except for the usual assumption of no downstream feedback.
- No ambiguity exists in the decode procedure at Mach = 1. Thus mixed flows can be treated with no a priori knowledge of the relative velocity magnitudes.
- Solid wall boundaries can be handled in the QP method with no-slip values. Regular parabolic procedures must avoid these type boundaries.
- Inclusion of more than one upstream plane will allow second order accuracy to be maintained in the quasi-marching coordinate.
- The QP scheme can accommodate either explicit or implicit "time" relaxation finite-differences.
- Within the basic framework of the QP scheme, an elliptic region could be treated before, during or after a marching integration simply by including k-data planes (instead of 2) during the relaxation.
- The QP algorithm requires very little addition storage over a classical parabolic method; and requires many times less storage than a fully elliptic treatment. Thus on STAR, the GIM/QP code could march out very large flow fields with no large page faults.
- By dropping the cross-plane viscous terms, the QP procedure becomes Quasi-hyperbolic with no further coding changes. Thus one algorithm can accommodate either parabolic or hyperbolic flows.

4.4 RESULTS OF COMPUTATION

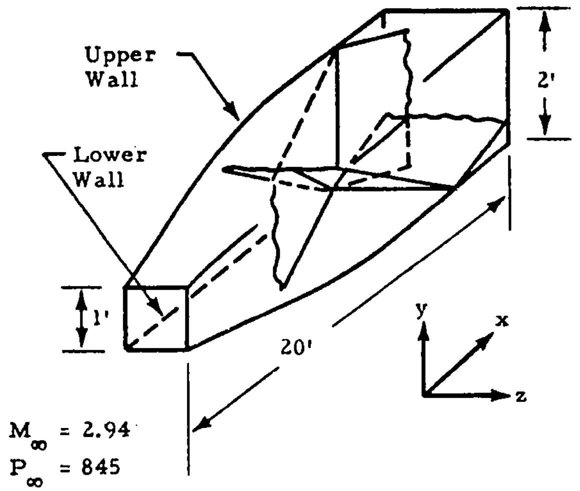
The QP code has been essentially completed and a number of test cases exercised. Three test problems are shown in this section for illustration of the Quasi-Parabolic code. These cases are:

1. Flow in a three-dimensional duct with an expansion-recompression and interaction of two shock waves.
2. Flow over a 10 degree planar wedge.
3. Two-dimensional viscous flow resulting from interaction of a nozzle exhaust with a supersonic freestream.

Other cases are currently in progress, including a boundary layer calculation, containing subsonic and supersonic flow.

The first problem shown is depicted in Fig. 4-4. The 1 x 1 square nozzle expands via a trignometric variation to 10 units and then has a constant 2 x 2 cross section. The supersonic Mach number expands up to the 10 unit plane then, due to recompression, shock sheets form at the top and outer side wall. The two shocks intersect as depicted in the figure. The QP code was used essentially in a hyperbolic mode with free slip inviscid solid walls and contained approximately 24,000 grid points. The purpose of the case is to determine the ability of the QP marcher to handle rapid expansions and strong compressions (shock capturing).

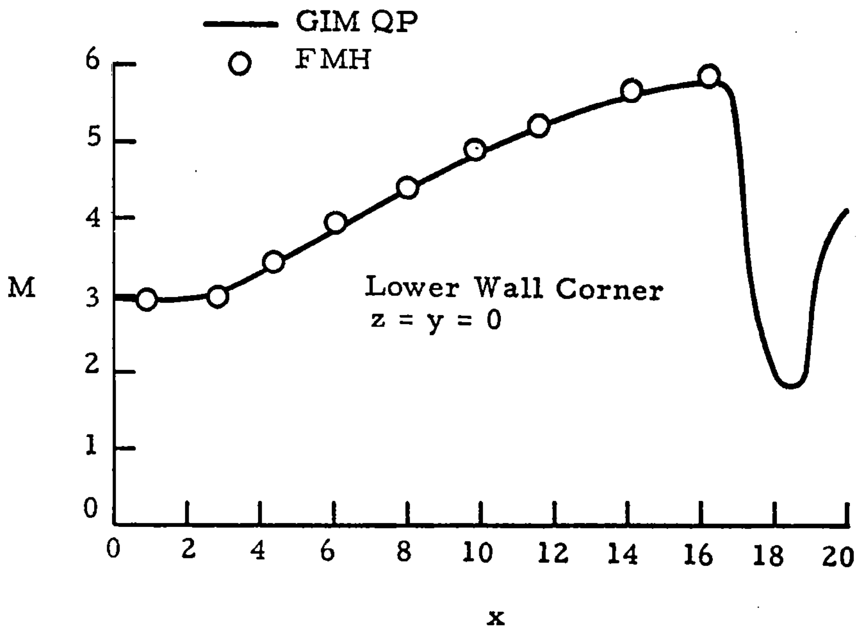
The solution for Mach number at the lower wall corner is shown on Fig. 4-4. A comparison is attempted here with a forward-marching hyperbolic code of the classical variety (a MacCormack code). The GIM/QP solution shows a strong shock wave while the other marcher would not solve for the large gradients at all. Figure 4-5 shows additional profiles for this case. The pressure ratio (local to inlet) is shown for both the upper and lower wall corners. Comparison is made to a published solution (Ref. 11). Excellent agreement is seen for the smooth upper wall profiles and for the expansion portion of the lower wall corner. At the axial location where the shock intersection occurs, the two solutions differ considerably. The GIM/QP code, using a <u>first order</u> finite difference scheme agreed very well with the ATL results.



$(z, y) = 1.5 - 0.5 \sin(\frac{\pi}{2} + \frac{\pi x}{10}), x \leq 10$

$(z, y) = 2, x > 10$

Configuration



Mach Number vs Axial Distance on "Lower Wall" Corner

Fig. 4-4 - Three-Dimensional Square Duct GIM Hyperbolic Computation.

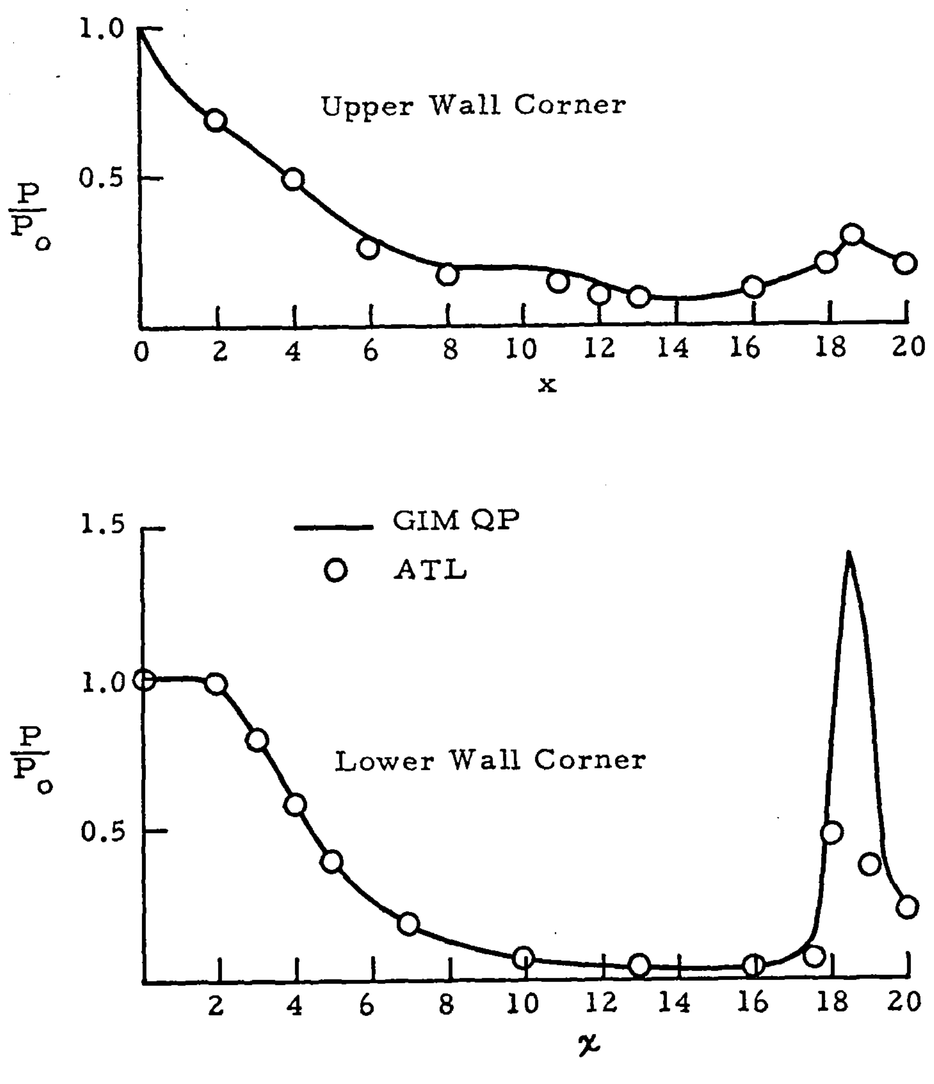


Fig. 4-5 - Three-Dimensional Square Duct GIM Hyperbolic Computation (Axial Pressure Distribution).

However, the second order QP algorithm produced the curve shown in Fig. 4-5, i.e., a larger pressure rise. As a check on the accuracy of the QP shock wave, 15,000 grid points were placed between 16 and 20 units. Very similar results were obtained as with the coarser mesh (11 x 11 x 81). It is thus felt that the GIM QP code is calculating the correct pressure rise across the shock.

In order to test the shock-capture capabilities of the QP finite difference scheme, an oblique shock on a 10-degree two-dimensional wedge was computed. Two example cases were run with incident Mach numbers of 1.8 and 2.4. The same 60 x 51 node grid was used for both calculations. Each case required about 26 seconds to converge. The results are shown in Fig. 4-6 as the pressure ratio through the shock as a function of vertical position and pressure rise from the NACA 1135 shock tables. The shock was characteristically smeared over five grid points. The excellent agreement indicates a good shock-capturing capability with the QP second order backward difference scheme.

Case three consists of a parabolic, viscous flow in the configuration of Fig. 4-7. A nozzle with high pressure exhausts into a lower pressure, hypersonic freestream flow. This case was solved with the GIM elliptic code with 940 nodes and reported in Ref. 2. The QP algorithm gives virtually identical results as given by the full Navier-Stokes code. The grid used and the steady state Mach and pressure contours are shown on Fig. 4-7. Comparison of this solution with the reported values of Ref. 2 and with the inviscid SEAGULL code of Ref. 12 are shown in Figs. 4-8 and 4-9. The SEAGULL is an inviscid, slip-line, shock fitting, forward marching code. Figure 4-8 shows vertical pressure distributions at three axial stations in the shear region, and Fig. 4-9 gives the corresponding Mach number plots. As seen by the comparisons, the GIM marching algorithm does indeed work as expected and gives quantitatively the same answers as the other codes. Application to a boundary layer problem is currently under way.

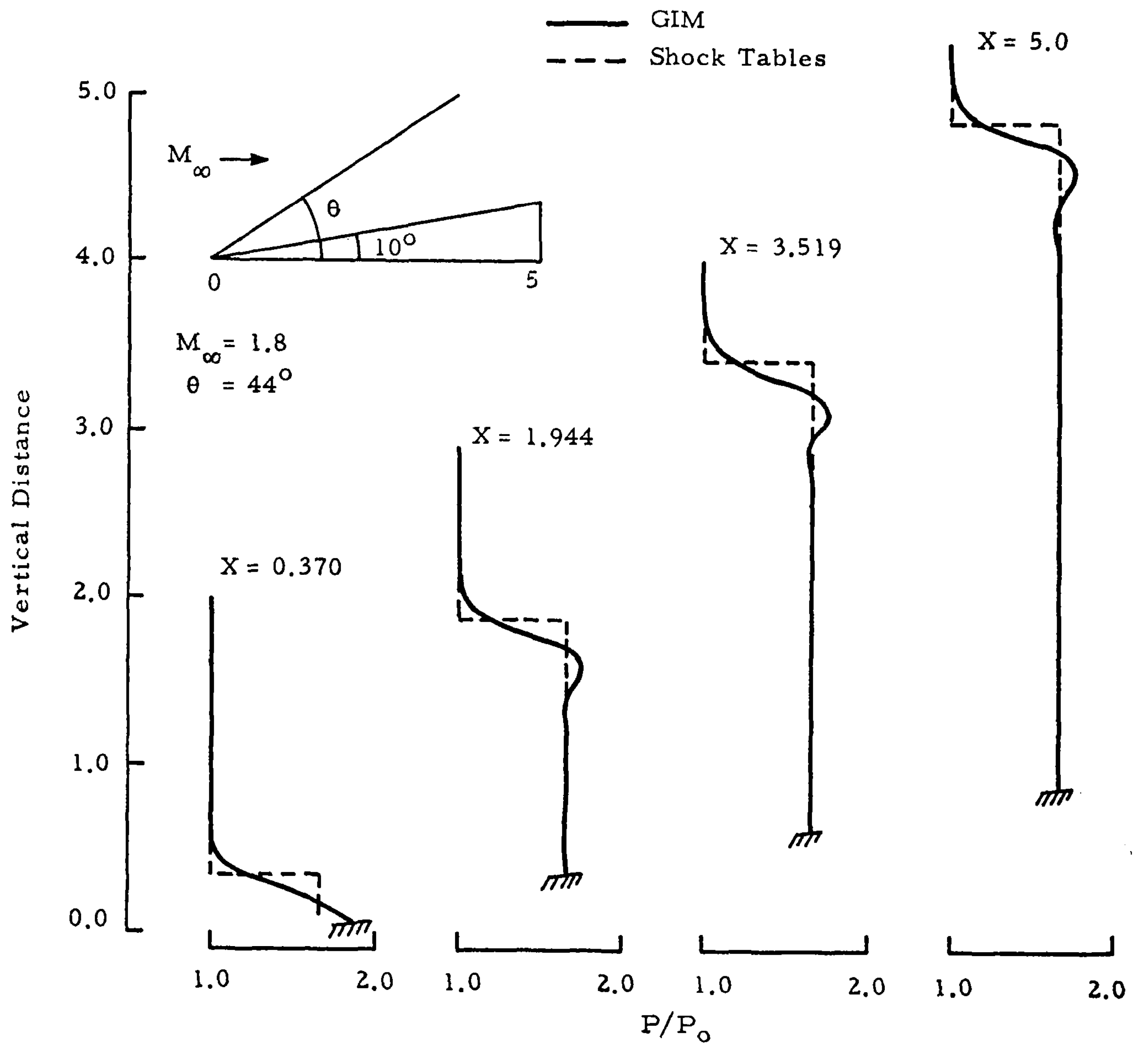


Fig. 4-6a - Wedge Shock Case to Verify Capture Technique in GIM/QP Code (M_∞ = 1.8, θ = 44 deg).

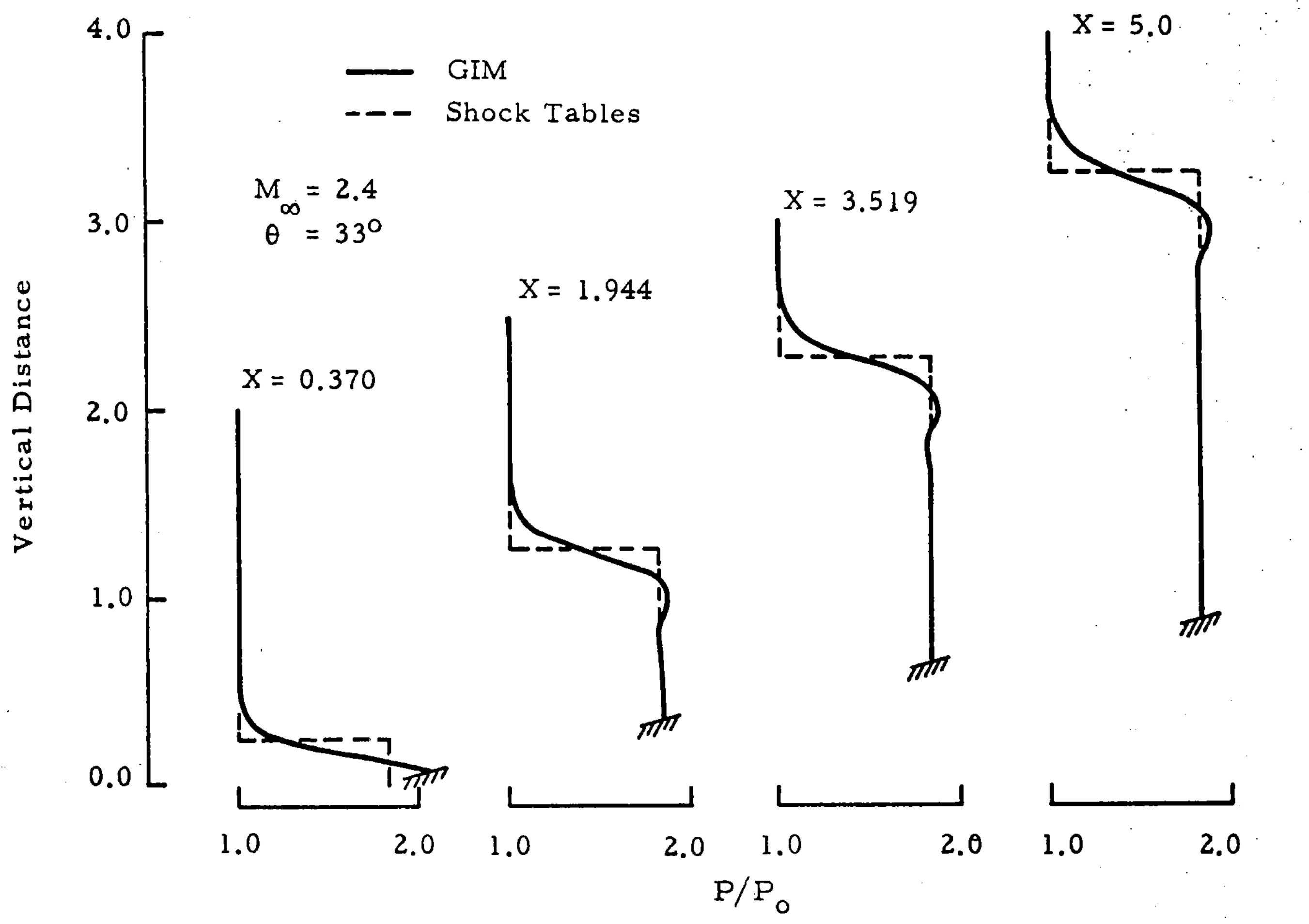
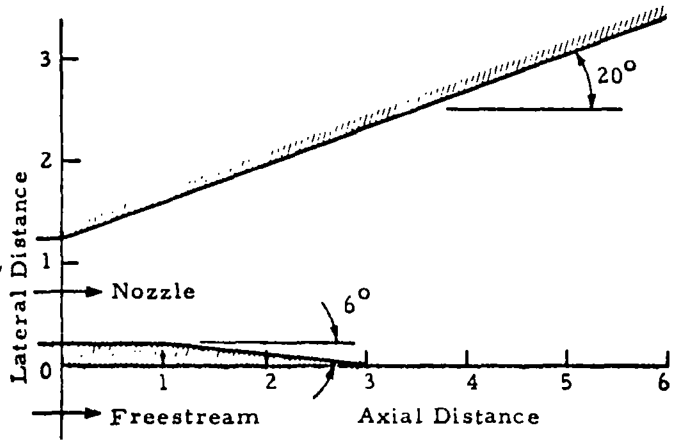
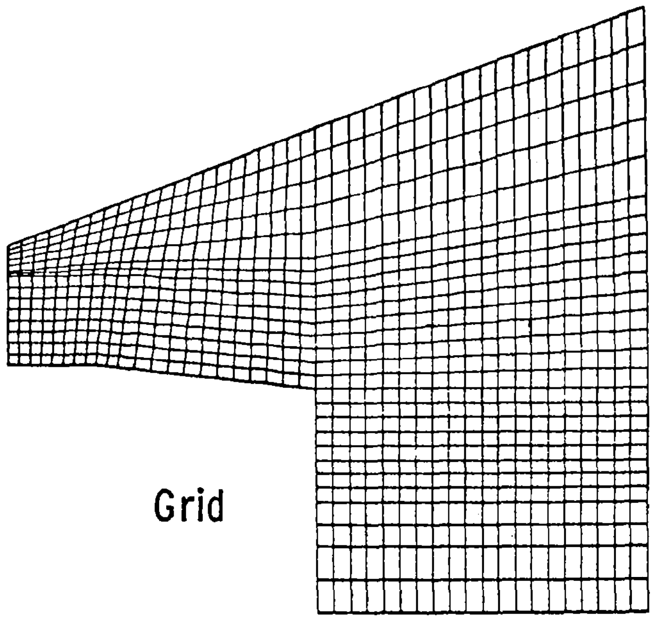


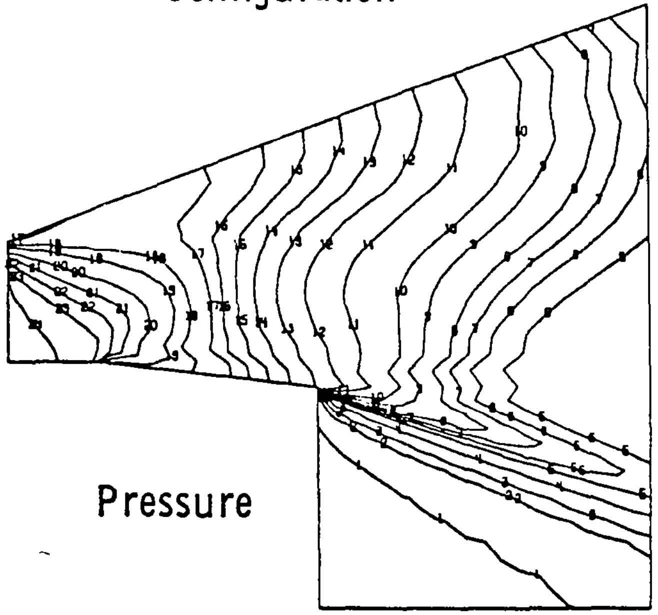
Fig. 4-6b - Wedge Shock Case to Verify Capture Technique in GIM/QP Code (M_∞ = 2.4, θ = 33 deg).



	γ	M	P
Nozzle	1.27	1.657	1924
Freestream	1.27	5.0	106

Configuration





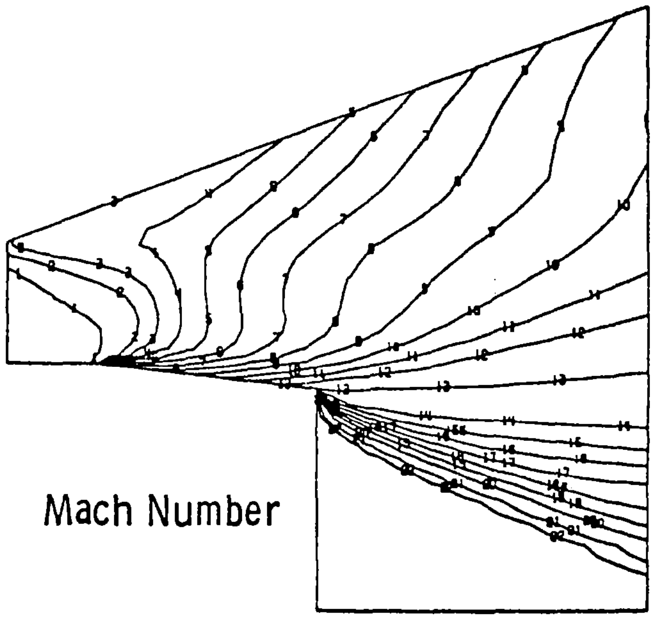


Fig. 4-7 - Quasi-Parabolic Code Applied to Shear Flow.

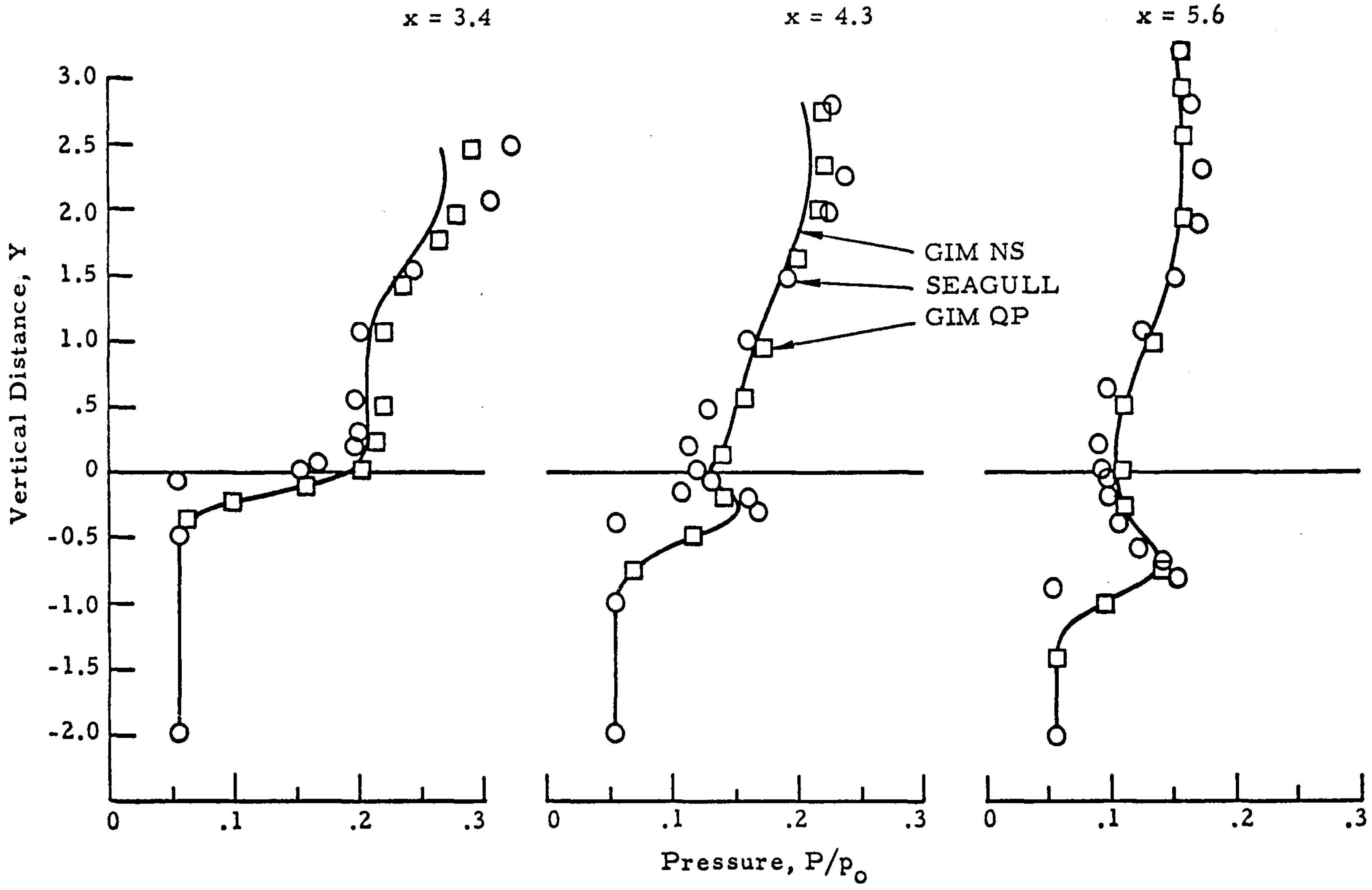


Fig. 4-8 - Quasi-Parabolic Shear Flow Computation (Vertical Pressure Distributions at Three Axial Stations).

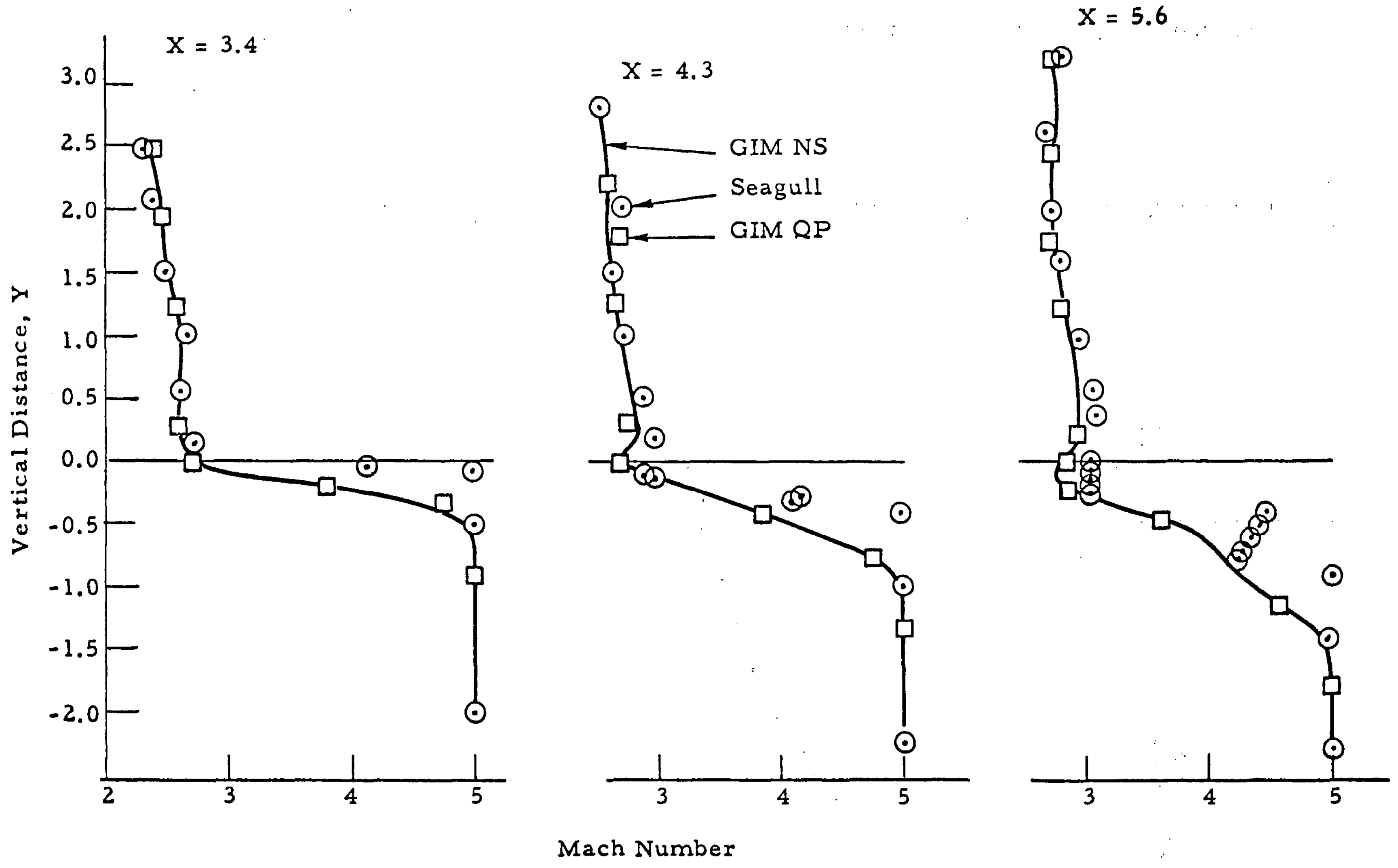


Fig. 4-9 - Quasi-Parabolic Shear Flow Computation (Vertical Mach Number Distributions at Three Axial Stations).

5. REFERENCES

1. Spradley, L. W., and M. L. Pearson, "GIM Code User's Manual for the STAR-100 Computer," NASA CR-3157, 1979.
2. Spradley, L. W., P. G. Anderson and M. L. Pearson, "Computation of Three-Dimensional Nozzle-Exhaust Flow Fields with the GIM Code," NASA CR-3042, 1978.
3. Beam, R. M., and R. F. Warming, "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Paper 77-645, Third Computational Fluid Dynamics Conference, Albuquerque, N. M., June 1977.
4. Salas, M. D., "Shock Fitting Method for Complicated Two-Dimensional Supersonic Flows," AIAA J., Vol. 14, No. 5, May 1976.
5. Roberts, D. W., and C. K. Forester, "Parabolic Procedure for Flows in Ducts with Arbitrary Cross Section," AIAA J., Vol. 17, No. 1, January 1979.
6. Patankar, S. V., and D. B. Spalding, "A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flow," Int. J. Heat Mass Trans., Vol. 15, 1972, pp. 1787-1806.
7. Rubin, S. G., and T. C. Lin, "A Numerical Method for Three-Dimensional Viscous Flow; Application to the Hypersonic Leading Edge," J. Comp. Phys., Vol. 9, 1972, pp. 339-364.
8. Hirsh, R.S., "Calculation of Supersonic Three-Dimensional Free-Mixing Flows Using the Parabolic-Elliptic Navier-Stokes Equations," Aerodynamic Analyses Requiring Advanced Computers, Part I, NASA SP-347, 1975, pp. 543-565.
9. Lubard, S. C., and W.S. Helliwell, "Calculation of the Flow on a Cone at High Angle of Attack," AIAA J., Vol. 12, No. 7, 1974.
10. Lin, T. C., and S. G. Rubin, "A Numerical Model for Supersonic Viscous Flow over a Slender Reentry Vehicle," AIAA Paper No. 79-0205, New Orleans, January 1979.
11. Dash, S. M., and P. D. Del Guidice, "Numerical Methods for Calculation of Three-Dimensional Nozzle Exhaust Flow Fields," Aerodynamic Analyses Requiring Advanced Computers, NASA SP-347, Part I, Mar. 1975, pp. 659–679 .
12. Salas, M. D., "Shock Fitting Method for Complicated Two-Dimensional Supersonic Flows," AIAA J., Vol. 14, No. 5, May 1976.

Appendix A
USE OF THE GIM SE-2 CODE

by

L. W. Spradley

Appendix A

A.1 THE GIM/STAR SE-2 CODE

The Blue Book (Ref. 1) describes the version of the GIM/STAR code designated SE-1 (STAR Elliptic Version 1). This reference manual contains input guides and user information for the code. Since the publication of this Blue Book there have been a number of changes to the code which have not been documented. Some of these changes were necessary to allow large problems to be run with a minimal number of large page faults while others were made to reduce the possibility of wasting computer time generating MATRIX analogs on a bad grid.

The input changes are not extensive but the user should use this Appendix in conjunction with the Blue Book when running a GIM/STAR problem. The following subsections describe the changes for the program modules and file usage.

A.2 GEOM MODULE

Module 1 of the GIM SE-1 deck was titled GEOMAT as it contained both the geometry and grid generation and the matrix coefficient assembly. The SE-2 version has the two operations broken out into separate modules. The first module of SE-2 is titled GEOM, as it now only performs the geometric description and grid generation (see Fig. A-1).

The user should be aware of the differences in this module between versions 1 and 2:

- Input cards 16 and 17 (in the Blue Book) are no longer used – just omit cards 16 and 17.

Card Type	Parameter List/Format
1	HEADER(I), I = 1, 72 (12A6)
2	NZONES, IDIM, ISTEP, IMATRX, IMATE (5I5)
3	IWRITE, LWRITE, NWRITE (3I5)
4	KC(I), I = 1, 6 (6A5)
5	NSECTS (I5)
6	MAPE(I), I = 1, 12 (12I5)
7	MAPS(I), I = 1, 6 (6I5)
8	(IBWL(I), I = 1, 6), ITRAIN (7I5)
9	(NNOD(I), I = 1, 3), (ISTRCH(I), I = 1, 3) (6I5)
10	DIVPI(I), I = 1, 3 (3E10.4)
11	[AETA(J, I), I = 1, NNOD(J)], J = 1, IDIM (8E10.4)
12	[(AC(I, K, J), I = 1, 8), J = 1, 4 or 12], K = 1, 5 (8E10.4)
13	[AS(I, J), I = 1, 8], J = 1, 6 (8E10.4)
14	(PT(I, J), I = 1, 5), J = 1, 4 or 12 (8E10.4)
15	[(PMAX(I, K, J), I = 1, 5), ETAMAX(K, J), K = 1, 4], J = 1, 4 or 12 (6E10.4)

Fig. A-1 - Input Guide for GEOM Module (SE-2 Code).

- File 17 is not output from the geometry module, rather File 18 is now to be saved. This new File 18 is to be subsequently input to the new MATRIX module.
- Card Type 4 has been changed. The values of α are no longer input, but rather a set of flags to retrieve the correct α's are now used. The parameter KC is set to alphabetic characters F, B, etc., for forward or backward differences. KC(1) is for x step 1, KC(2) for y step 1, etc., through KC(6) which is z step 2. The format is 6A5.

A.3 MATRIX MODULE

The new MATRIX module of the SE-2 code now performs the analog coefficient calculation and file creation. This module should be executed following GEOM and before INTEG. The File 18 which was output from GEOM is now input to the MATRIX module. File 17 needed by INTEG is to be saved from MATRIX. Figure A-2 gives the storage requirements for MATRIX.

The input to the MATRIX module consists of the Cards 16 and 17 which were omitted from the GEOM module, plus <u>one</u> new card. Each of these cards is now described:

Card	Parameters	Format
1	NDX, NDY, NDZ, ISNOPT	(4I5)
2	KC(I), I = 1,6	(6A5)
3	N1, IC, NT	(3I5)

<u>Card Type 1</u> Format (4I5)

Same as Card Type 16 (GEOM SE-1) p. 4-27

NDX	nodal decrement in the η_1-coordinate system
NDY	nodal decrement in the η_2-coordinate system
NDZ	nodal decrement in the η_3-coordinate system
ISNOPT	special node treatment flag

If ISNOPT = 1 the MATRIX module will calculate the number of special node terms placed on File 17 for input to INTEG

Matrix	2D	3D
/ACOM/	50*NN	196*NN
/PCOM/	4*NN	8*NN
(Q3MAP/	24*NN + 18*NSPEC + 65542	48*NN + 18*NSPEC +65542
/IRFBC/	6*NSPEC	6*NSPEC
/JCFBC/	6*NSPEC	6*NSPEC
/PAFBC/	6*NSPEC	6*NSPEC

where

NN	=	total number of nodes
NSPEC	=	number of special node terms allowed for in DIMENSION statements (DYNMAT input)

The common block sizes may be calculated for each problem size to determine the ideal grouping on the LOAD card. If in doubt assign each block to a new large page boundary as below. Do not use GRLPALL, but use

GRLP=*ACOM, GRLP=*PCOM, GRLP=*Q3MAP, GRLP=IRFBC,

etc.

Fig. A-2 - Module Common Block Sizes.

If ISNOPT = 0	the entire array of special node terms will be placed on File 17. The size of the array is determined by DYNMAT input (NSPEC parameter).

<u>Card Type 2</u> Format (6A5) Analog Choice Card

This card consists of a sequence of six characters (F or B) identifying the difference direction (forward or backward) for X, Y, and Z Step 1 and X, Y, Z Step 2, respectively.

Examples:

F F F B B B

forward, forward, forward, backward, backward, backward for three-dimensional problems and

F B _ B F _

forward, backward, backward, forward for two-dimensions.

This is a new card for version SE-2 and is identical to GEOM card type 4.

<u>Card Type 3</u> Format (3I5)

Nodal analog print control card.

N1	first node of a print sequence
IC	print increment
NT	total number of nodes to print for this sequence.

(See page 4-29 SE-1 manual for complete description.)

Any number of cards of this type may be input.

Place a -1 in Columns 4 and 5 on last card to terminate.

<u>Dynamic Dimension for MATRIX</u>

The new MATRIX module has its own dynamic dimensioning sequence. The same deck DYNMAT is used for MATRIX and GEOM, but the input of a third parameter is optional in MATRIX.

The DYNMAT deck is to be executed before running the MATRIX deck.

The input consists of one card:

Card	Parameters	Format
1	NN, IDIM, NSPEC	(3I5)

The definition of these input variables are the following:

Card 1 Format (3I5)

NN	number of nodes
IDIM	dimensionality (2 or 3)
NSPEC	number of special node terms to allow for in DIMENSION statements. If left blank, or zero, the arrays will be dimensioned to NN. The actual number of special node terms will be calculated and printed out in MATRIX. This value is then input to INTEG. (Not used by GEOM.)

A.4 INTEG MODULE (SE-2)

This module has remained virtually intact from a user standpoint. Three additional options have been added since the Blue Book was issued. These are:

- Capability to compute a CFL time step automatically over a multi-zoned grid.
- Treatment of downstream subsonic boundary conditions using a mass balance condition. This option was added under another NASA contract and is documented here for completeness.
- Input of a set of flags denoting the finite-difference direction. This aids in a more complete set of difference options and allows for full vectorization of all schemes.

Figure A-3 is a summary Input Guide for the INTEG SE-2 module. Note that the new input cards are designated 2a, 2b, 2c, 3a and 6a. All except 6a are optional and existing data decks will still work as they did for version SE-1. Card 6a must now be input in version SE-2. Card 2 has two additional inputs, IDS, IBOUND which control the optional input of 2a, 2b, and 2c. Zero values for the parameters on Card 2 signifies omission of the remaining Cards Type 2a, 2b and 2c. Figure A-3 is a description of the available options and each parameter that is to be input.

Figure A-5 describes each parameter that is input on the optional Cards 2a, 2b and 2c. An example of the use of the subsonic boundary condition option is shown in the sample grid of Fig. A-6.

The time step calculation option is controlled via the value of KZONES read as the last data on Card 3. If this is omitted (=0), then one zone is assumed. If KZONES > 0, then this signals the code that a multiple zone problem is being run. In this case, the value of KZONES should be equal to the number of zones used in the geometry module. If KZONES = 0, then Card Types 3a are <u>not</u> used, but any value of KZONES > 0, requires the input of come Cards 3a. The number of Cards 3a to be input is equal to KZONES-1. The time step information for zone number 1 is input on Card 3 itself. Figure A-7 describes the input of this time step information.

Card 6a is simply the KC values used in GEOM and MATRIX, i.e.,

F F F B B B

in format 6A5. This card must agree with the previous module's usage.

One additional Fig. A-8 is included in this subsection. This chart shows formulas for determining the COMMON block sizes for the INTEG module. These values are needed for large problems to set up the LOAD card as described in the Blue Book (Ref. 1).

Card Type	Parameter List/Format
1	ICASE, IITITLE(I), I = 1, 78) (I2, 78A1)
2	IDIM, METHOD, ITMAX, IPRNT, ITSAVE, ISTART, IOTYPE, IUNITS, ITSTRT, IVISC, IDIST, ISPEC, IDS, IBOUND (14I5)
2a	INFOUT, IJUMPO, JJUMPO, NIOUT, NJOUT, ICALC, AMFLW (6I5, E10.0)
2b	INFINL, IJUMPI, JJUMPI, NIIN, NJIN, ICALC, OUTMFL (6I5, E10.0)
2c	INFINL, IJUMPI, JJUMPI, NIIN, NJIN, INFOUT, IJUMPO, JJUMPO, NIOUT, NJOUT, ICALC (11I5)
3	NN, NNX, NDX, NNY, NDY, NNZ, NDZ, NPM, KZONES (9I5)
3a	KST, KNX, KDX, KNY, KDY, KNZ, KDZ (7I5)
4	DTIME, DTFAC, INCDT (2E10.0, I5)
5	REALMU, REALK, GAMS1, GAMS2, WM1, WM2, DK, RK (8E10.0)
6	EMU, ELAM, ERHO, ESPEC (4E10.0)
6a	KC(I), I = 1, 6 (6A5)
7	NNPM(I), NCPM(I), (NNCPM(I, J), J = 1, 5), ANGPM(I); I = 1, NPM (7I5, E10.0)
8	(NCT(I, J, K), PXPM(I, J, K), PYPM(I, J, K), K = 1, 4); J = 1, NCPM(I); I = 1, NPM (I5, 2E10.0)
9	RHOZ, PZ, ASTAR, NINC, A, B (3E10.0, I5, 2E10.0)
10	NJ, INC, NTOT, ITAN, ITYPE (5I5)
11	RI, UI, VI, WI, PI, CSI (6E10.0)
12	N1, IC, NT (3I5)

Fig. A-3 - Input Guide for INTEG Module (SE-2 Code).

Card	Col.	Format	Variable	Description
Type 2	1-5	I5	IDIM	See Blue Book
	6-10	↓	METHOD	↓
	11-15	↓	ITMAX	↓
	16-20	↓	IPRNT	↓
	21-25	↓	ITSAVE	↓
	26-30	↓	ISTART	↓
	31-35	↓	IOTYPE	↓
	36-40	↓	IUNITS	↓
	41-45	↓	ITSTRT	↓
	46-50	↓	IVISC	↓
	51-55	↓	IDIST	↓
	56-60	↓	ISPEC	↓
	61-65	↓	IDS	Boundary Condition Flag = 0, one-sided differences (supersonic) = 1, mass balance technique (subsonic)
	65-70	I5	IBOUND	Note: If IDS. Eq. 1 IBOUND should be set to either -1, 0, or 1. If IDS. Eq. 0, IBOUND is left blank = -1, input inlet mass flow and calculate exit mass flow = 0, input exit mass flow and calculate inlet mass flow = 1, calculate both inlet and exit mass flow

Note: If IDS. Eq. 1 card types 2a, 2b and 2c must follow type 2 card. The use of types 2a, 2b and 2c depends on the value of IBOUND.

If IBOUND = -1, use Type 2a

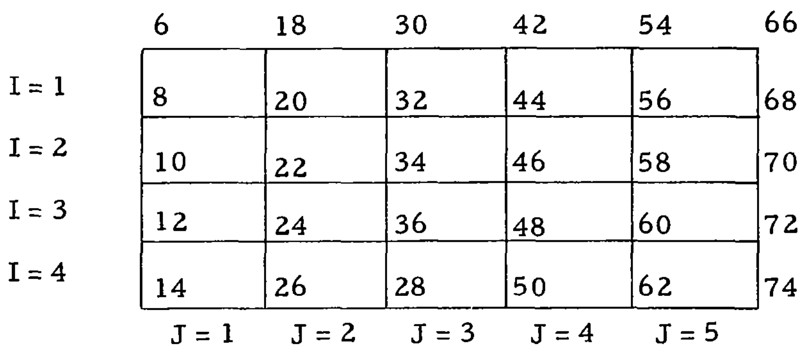
If IBOUND = 0, use Type 2b

If IBOUND = 1, use Type 2c

Fig. A-4 - Definition of Parameters for Card 2.

Card	Col.	Format	Variable	Description
Type 2a	1-5	I5	INFOUT	Starting node on exit plane
	6-10	I5	IJUMPO	Nodal increment in ith direction on exit plane
	11-15	I5	JJUMPO	Nodal increment in jth direction on exit plane
	16-20	I5	NIOUT	Number of elements in ith direction on exit plane
	21-25	I5	NJOUT	Number of elements in jth direction on exit plane
	26-30	I5	ICALC	Velocity update flag = 1, update inlet velocities = 2, update exit velocities
	31-40	E10.0	AMFLW	Inlet mass flow rate (input by user)
Type 2b	1-5	I5	INFINL	Starting node on inlet plane
	6-10	I5	IJUMPI	Nodal increment in the ith direction on inlet plane
	11-15	I5	JJUMPI	Nodal increment in the jth direction on inlet plane
	16-20	I5	NIIN	Number of elements in ith direction on inlet plane
	21-25	I5	NJIN	Number of elements in jth direction on inlet plane
	26-30	I5	ICALC	Velocity update flag (see Card Type 2a)
	31-40	E10.0	OUTMFL	Exit mass flow rate (input by user)
Type 2c	1-5	I5	INFINL	Starting node on inlet plane
	6-10	I5	IJUMPI	Nodal increment in the ith direction on inlet plane
	11-15	I5	JJUMPI	Nodal increment in the jth direction on inlet plane
	16-20	I5	NIIN	Number of elements in ith direction on inlet plane
	21-25	I5	NJIM	Number of elements in jth direction on inlet plane
	26-30	I5	INFOUT	Starting node on exit plane
	31-35	I5	IJUMPO	Nodal increment in the ith direction on exit plane
	36-40	I5	JJUMPO	Nodal increment in the jth direction on exit plane
	41-45	I5	NIOUT	Number of elements in ith direction on exit plane
	46-50	I5	NJOUT	Number of elements in jth direction on exit plane
	51-55	I5	ICALC	Velocity update flag (see Card Type 2a)

Fig. A-5 - Description of Input Parameters for Optional Card Types 2a, 2b, 2c (Subsonic Boundary Conditions).



Example:

IBOUND = 0   INFINL = 6, IJUMPI = 2, JJUMPI = 12, NIIN = 4, NJIN = 5

Fig. A-6 - Example of Subsonic Boundary Condition Usage.

Card	Col.	Format	Parameter	Description
3	5	I5	NN	
	10	I5	NNX	
	15	I5	NDX	
	20	I5	NNY	See Blue Book
	25	I5	NDY	
	30	I5	NNZ	
	35	I5	NDZ	
	40	I5	NPM	
	45	I5	KZONES	

The number of zones that was used to construct the grid. This is used to allow a CFL time step to be computed over multiple zones. Set to 1 for a single zone problem.

Card	Col.	Format	Parameter	Description
3a	5	I5	KST	Starting node number of this zone.
	10	I5	KNX	Number of nodes in η_1 direction for this zone
	15	I5	KDX	Nodal decrement in η_1 direction for this zone
	20	I5	KNY	Number of nodes in η_2 direction for this zone
	25	I5	KDY	Nodal decrement in η_2 direction for this zone
	30	I5	KNZ	Number of nodes in η_3 direction for this zone. Set to 1 for 2-D flow.
	35	I5	KDZ	Nodal decrement in η_3 direction for this zone. Set to 1 for 2-D flow.

Note: Input Card Type 3a for each multiple zone to be used in computing a CFL time step. The number of Cards 3a is equal to KZONES-1, where KZONES is input on Card 3.

Fig. A-7 - Description of Parameters for Optional Card Type 3a Input.

Common Block Names	Axisymmetric		2-D		3-D	
	1 Gas	2 Gases	1 Gas	2 Gases	1 Gas	2 Gases
/PRIM/	5*NN+1	6*NN+1	5*NN+1	6*NN+1	6*NN	7*NN
/TAU/	9*NN+3	9*NN+3	9*NN+3	9*NN+3	12*NN	12*NN
/TMVEC/	2*NN	2*NN	2*NN	2*NN	2*NN	2*NN
/VPROP/	2*NN	4*NN	2*NN	4*NN	2*NN	4*NN
/VBUF/	8*NN	10*NN	8*NN	10*NN	10*NN	12*NN
/BOUND/	5*NB+1	5*NB+1	5*NB+1	5*NB+1	5*NB+1	5*NB+1
/EBUF/	8*NN+4	10*NN+5	8*NN+4	10*NN+5	15*NN	18*NN
/XBUF/	7*NN+4	7*NN+4	7*NN+4	7*NN+4	10*NN+1	10*NN+1
/STEP/	3*NN+10	3*NN+10	3*NN+10	3*NN+10	4*NN+9	4*NN+9
/AXSYM/	8*NN	9*NN	8	9	9	10
/Q3MAP/	24*NN+18*NSPEC +6+COMP		24*NN+18*NSPEC +6+COMP		48*NN+18*NSPEC +6+COMP	

NN	=	total number of nodes.
NB	=	number of boundary nodes.
NSPEC	=	number of special nodes.
COMP	=	amount of storage need to complete a large page.

Fig. A-8 - INTEG Module Common Block Sizes for SE-2 Code.

A.5 GIM SE-2 FILE DESCRIPTIONS

Following is a brief description of the files used in the STAR SE-2 system. In all but very small problems setups, (a few hundred nodes), a REQUEST card must be used for each file. The form of the REQUEST card is as follows:

REQUEST (FILEXX/NSPGS, T = P)

where

NSPGS = the number of small pages of disk space allocated to the file

1 small page = 512 words

Formulas for calculating NSPGS are now given for each file.

FILE 16 GEOM

Work file used in GEOM only (Binary)

NSPGS ≈ 15*NN/512 NN = no. of nodes

FILE 17 MATRIX/INTEG (Binary)

Nodal analog file created in MATRIX and used in INTEG.

2D

NLPGS = (16*NN + 18*NSP + 6)/65536 rounded up to next whole integer

NSPGS = NLPGS*128 + 1

3D

NLPGS = (48*NN + 18*NSP + 6)/65536 rounded up to next whole integer

NSPGS = NLPGS*128 + 1

where

NN = total number of nodes

NSP = number of special node terms.

FILE 18 GEOM/MATRIX (Binary)

File containing matrix assembly data. Created in GEOM and used in MATRIX.

2D

NSPGS = 50*NN/512 + 1

3D

NSPGS = 196*NN/512 + 1

FILE 20 GEOM/INTEG/GIMTEK (Formatted)

Nodal geometry file created in GEOM and used in INTEG and GIMTEK

2D

NSPGS ≈ 14*NN/512

3D

NSPGS ≈ 20*NN/512

FILE 22 INTEG/GIMPLT (Formatted)

Flowfield solution file created in INTEG and used both as a restart file and in GIMTEK.

2D

NSPGS ≈ 11*NN/512 per record

3D

NSPGS ≈ 14*NN/512 per record

Multiply by the number of iteration increments saved.

Controllee File Sizes

The size of the controller file is specified on the LOAD card in small pages. Formulas are given below for calculating the size required for a given problem.

GEOM

$$\text{NLPGS} = \begin{cases} 50{*}\text{NN}/65536 + 3 & \text{2D} \\ 196{*}\text{NN}/65536 + 3 & \text{3D} \end{cases}$$

rounded up to next whole integer

NSPGS = NLPGS*128 This is the value that goes on the LOAD card.

MATRIX

No single formula exists to calculate the controllee file size for the MATRIX module. The procedure is to calculate the number of large pages (65536 words each) required for each GRLP parameter on the LOAD card, add these up, add 2 for other storage and multiply the result by 128.

INTEG

The same rule applies to the INTEG controllee file size as to the MATRIX module. Use the common block sizes to compute the number of large pages, add them up, add a couple and then multiply by 128 to get the controllee file size number.

A.6 PLOT MODULE (GIMTEK)

The GIM SE-2 code plotting module is now titled GIMTEK. This reflects the modifications which were made to the GIMPLT SE-1 module in order to use the Tektronix 4014 for graphic output. The user need not be aware of the internal program changes that were made. The input data are identical to the SE-1 version. Three items of significance to the user are now described:

$$\boxed{KMAX_{10} = \max \begin{Bmatrix} 5040_{10} + 8*NN \\ 11*NN \end{Bmatrix} \qquad NN = \text{number of nodes}}$$

The CM field length specified on the job card is calculated by

$$\boxed{CM_{10} = KMAX_{10} + 23000_{10}}$$

Notes: 1. This parameter must be set in the program and the array "A" dimensioned to this value.

2. CM must be converted to octal for specification on job card, and RFL card

3. <u>Example</u>

$$NN = 2000$$

$$KMAX_{10} = \max \begin{Bmatrix} 5040 + 8*2000 \\ 11*2000 \end{Bmatrix} = \max \begin{Bmatrix} 21040 \\ 22000 \end{Bmatrix}$$

$$= 22000$$

$$CM_{10} = 22000 + 23000 = 45000_{10}$$

use $CM = 130000 \; (130000_8 = 45056_{10} > 45000_{10})$

Fig. A-9 - GIMTEK Core Requirements.

- The formulas on page 6-25 of the Blue Book for computing core sizes for the plot module are no longer valid. Figure A-9 gives the revised formulas and an example calculation.
- The plot save command was changed on the software system. The new save name is

  SAVPVF.

- The routine that we use for obtaining GIMTEK plots from the Tektronix 4014 is

  PLIST.

  This allows enough options to select only those plots needed and also allows an unlimited time to examine a plot before proceeding.

The input data for GIMTEK is the same as described in the Blue Book. Figure A-10 is a summary of the required input data presented here for completeness. The user is referred to the Blue Book for a definition of the parameters.

A.7 EXAMPLE RUNSTREAMS FOR THE SE-2 CODE

The following pages show example runstreams that have been used for the SE-2 code on the STAR-100 machine. These should aid the new user in setting up a deck for GIM SE-2:

Fig. A-11 - GEOM Module Only
Fig. A-12 - MATRIX Module Only
Fig. A-13 - GEOM/MATRIX Combination Run
Fig. A-14 - INTEG Run Only
Fig. A-15 - GIMTEK Run

Note: The "blanks" which show up on the card listings are 7-8-9 punch cards, i.e., end of record.

The files for GIM SE-2 are cataloged under user number 838700C as GEOMB, MATRIXB, INTEGB and GIMTEKB.

Card Type	Parameter List/Format
1	ITITLE(1), ITITLE(2)
	(2A40)
2	NX, ITERAD, ITRBLK, KDIM, ISP
	(5I5)
3	GAMMA, FACTOR, RK, PO, TO, RHOO
	(6E10.0)
Specs.	
S-1	NPLT, STITLE, IVIEW, ISYM, ITHET1, IAXIS1, ITHET2, IAXIS2, IXTABL, VFAC
	(I5, 5X, A20, 8I5, E10.0)
S-2	NTYPE, JO, IJUMP, JJUMP, NI, NJ, IPRNT
	(7I5)
Grid	
G-1	'GRID', IOPT, ICSCLE, NSPECS, (ISPEC(I), I = 1, NSPECS)
	(A4, 1X, I5, 25X, 2I5, 7I5)
G-2	(ISPEC(I), I = 8, NSPECS) if NSPECS > 7)
	(45X, 7I5)
VVEC	
V-1	'VVEC', IOPT, NITER, ICSCLE, NSPECS, (ISPEC(I), I = 1, NSPECS)
	(A4, IX, 2I5, 20X, 2I5, 7I5)
V-2	(ISPEC(I), I = 8, NSPECS) (if NSPECS > 7)
	(45X, 7I5)
I-1	(ITER(I), I = 1, NITER)
	(16I5)
Contours	
C-1	ITYPE, IOPT, NITER, NC, ITABLE, INCR, ICSCLE, NSPECS, ISPEC(I), I = 1, NSPECS)
	(A4, 1X, 5I5, 5X, 2I5, 7I5)
C-2	(ISPEC(I), I = 1, NSPECS) (if NSPECS > 7)
	(45X, 7I5)
I-1	(ITER(I), I = 1, NITER)
	(16I5)
L-1	(CVAL(I), I = 1, NC)
	(8E10.0)

Fig. A-10 - GIMTEK Summary Input Guide.

```
GEODCT,CM60000,T100.
USER,838700C.
CHARGE,101857,LRC.
GET(OLDPL=GEOM/UN=838700C)
GET(DYNMAT=DYNMAT/UN=838700C)
UPDATE(F,C=TAPE8)
DYNMAT.
TOSTAR(INPUT,TAPE3)

*ID MODS

 2541      3

STORE 838700 400SDS TESTDECK B
STRSIDE,T100.
FORTRAN(I=TAPE3,B=GEOMB,O=LB)
REQUEST(FILE16/75,T=P)
REQUEST(FILE18/974,T=P)
REQUEST(FILE20/100,T=P)
LOAD(GEOMB,CN=GEOMGO,1408,GRLPALL=  )
GEOMGO.
TOAS(Z=838700C,FILE18=BI,FILE20)

***    GEOMETRY DATA   *****
```

Fig. A-11 - GEOM Runstream.

```
MATRIX,CM60000,T100.
USER,838700C.
CHARGE,101857,LRC.
GET(OLDPL=MATRIX/UN=838700C)
GET(DYNMAT=DYNMAT/UN=838700C)
UPDATE(F,C=TAPE8)
COPYSBF(TAPE8,OUTPUT)
REWIND(TAPE8)
DYNMAT.
ATTACH(FILE18=FILE18B)
TOSTAR(INPUT,TAPE3,FILE18=BI//,U)

*ID  NONE

 2541      3  1714

STORE  838700  400SDS  TESTDECK  B
STRSIDE,T100.
FORTRAN(I=TAPE3,B=MATRB,O=LB)
REQUEST(FILE17/385,T=P)
LOAD(MATRB,CN=MATRGO,1920
,GRLP=*ACOM,GRLP=*Q3MAP,GRLP=*PCOM,*IRFBC,*JCFBC,*PAFBC)
MATRGO.
TOAS(Z=838700C,FILE17=BI)

   121     11      1      1
     F      F      F      B      B      B
     1      1     20
    -1
```

Fig. A-12 - MATRIX Runstream.

```
GEOMAT,CM60000,T100.
USER,838700C.
CHARGE,101857,LRC.
GET(OLDPL=GEOM/UN=838700C)
GET(DYNMAT=DYNMAT/UN=838700C)
UPDATE(F,C=TAPE8)
DYNMAT.
COPYCF(TAPE3,GEOMC)
REWIND(GEOMC)
RETURN(OLDPL)
RETURN(TAPE3)
RETURN(TAPE8)
GET(OLDPL=MATRIX/UN=838700C)
UPDATE(F,C=TAPE8)
DYNMAT.
COPYCF(TAPE3,MATC)
REWIND(MATC)
RETURN(OLDPL)
RETURN(TAPE3)
RETURN(TAPE8)
TOSTAR(INPUT,GEOMC,MATC)

*ID MODS

 2541      3

*ID NONE

 2541      3 1714

STORE 838700 400SDS TESTDECK B
STRSIDE,T100.
REQUEST(FILE16/75,T=P)
REQUEST(FILE17/385,T=P)
REQUEST(FILE18/974,T=P)
REQUEST(FILE20/100,T=P)
FORTRAN(I=GEOMC,B=GEOMB,O=LB)
LOAD(GEOMB,CN=GEOMGO,1408,GRLPALL=  )
GEOMGO.
FORTRAN(I=MATC,B=MATRB,O=LB)
LOAD(MATRB,CN=MATRGO,1920
,GRLP=*ACOM,GRLP=*Q3MAP,GRLP=*PCOM,*IRFBC,*JCFBC,*PAFBC)
MATRGO.
TOAS(Z=838700C,FILE18=BI,FILE17=BI,FILE20)

****    GEOMETRY DATA   *******

****   MATRIX DATA   *******
```

Fig. A-13 - GEOM/MATRIX Runstream.

```
INTEGA,CM60000,T200.
USER,012839C.
CHARGE,102110,LRC.
ATTACH(FILE17=FILE17A)
ATTACH(FILE20=FILE20A)
GET(OLDPL=INTEG/UN=838700C)
GET(DYNDIM=DYNDIM/UN=838700C)
UPDATE(F,C=TAPE8)
DYNDIM.
COPYCF(TAPE3,INTEGX)
REWIND(INTEGX)
RETURN(OLDPL)
RETURN(TAPE3)
RETURN(TAPE8)
TOSTAR(INPUT,INTEGX,FILE20,FILE17=BI//U)

*IDENT   MODS

 1225      1      0  1759

STORE 012839 400SDS TESTDECK B
STRSIDE,T200.
FORTRAN(I=INTEGX,B=INTEGB/100,O=LB)
LOAD(INTEGB,CN=INTEGO,2000
,GRLP=*PRIM,*TMVEC,*VPRCP,*TAU
,*EBUF,*UBUF,*BOUND,*XBUF,*STEP
,GROL=*Q3MAP)
INTEGO.
TOAS(Z=012839C,FILE22)

*****     INTEG DATA   *********
```

Fig. A-14 - INTEG Runstream.

```
GIMTEK,CM120000,T400.
USER,012839C.
CHARGE,102110,LRC.
GET(OLDPL=GIMTEK/UN=492429C)
UPDATE(F)
FTN(I=COMPILE,L=0)
ATTACH(TAPE20=FILE20A)
ATTACH(TAPE22=FILE22E)
RFL(120000)
ATTACH(LIBFTEK,LRCGOSF/UN=LIBRARY)
LDSET(LIB=LIBFTEK/LRCGOSF,PRESET=NGINF)
LGO.
SAVE(SAVPVF=SCRJET)

*ID   KORCHG
*I    GIMPLT.744
        ISET=ISET+1
*D    GIMPLT.9
        COMMON A(14840)
*D    GIMPLT.15
        KMAX=14840

*****    GIMTEK DATA   *******
STOP
```

Fig. A-15 - GIMTEK Runstream.

Appendix B

THREE-DIMENSIONAL LBI SCHEMES FOR THE NAVIER-STOKES EQUATIONS

by

Jürgen Thoenes

Appendix B

B.1 INTRODUCTION

Algorithms are developed for the solution of the three-dimensional compressible Navier-Stokes equations in conservation form. This work represents an extension of the methodologies outlined by Beam and Warming (Ref. B-1) and Spradley (unpublished information) and familiarity with the cited literature is assumed. A time-dependent algorithm for the unsteady equations is developed first and then a spatial marching scheme for the three-dimensional parabolized steady equations is obtained. Algorithms for one- or two-dimensional problems are easily obtained by simply deleting appropriate terms from the equations.

B.2 THREE-DIMENSIONAL UNSTEADY ALGORITHM

The three-dimensional compressible conservation equations can be written in conservative form

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} =$$

$$= \frac{\partial}{\partial x}\left[V_{11}(U, U_x) + V_{12}(U, U_y) + V_{13}(U, U_z)\right]$$

$$+ \frac{\partial}{\partial y}\left[V_{21}(U, U_x) + V_{22}(U, U_y) + V_{23}(U, U_z)\right]$$

$$+ \frac{\partial}{\partial z}\left[V_{31}(U, U_x) + V_{32}(U, U_y) + V_{33}(U, U_z)\right] \tag{B.1}$$

where U is the vector of conserved variables and E, F, G and V_{ij} are flux vectors.

A generalized single-step temporal finite difference scheme for advancing the solution of Eq. (B.1), is the following.

$$\Delta U^n = \frac{\theta \Delta t}{1+\xi} \frac{\partial \Delta U^n}{\partial t} + \frac{\Delta t}{1+\xi} \frac{\partial U^n}{\partial t} + \frac{\xi}{1+\xi} \Delta U^{n-1} \tag{B.2}$$

where $U^n = U(n\Delta t)$ and $\Delta U^n = U^{n+1} - U^n$. (Terms of order Δt^2 and Δt^3 have been neglected, for simplicity.)

If Eq. (B.1) is solved for $\frac{\partial U}{\partial t}$ and inserted in Eq. (B.2), the resulting expression for ΔU^n is

$$\begin{aligned}
\Delta U^n = \frac{\theta \Delta t}{1+\xi} \Bigg[& \frac{\partial}{\partial x} (-\Delta E^n + \Delta V_{11}^n + \Delta V_{12}^n + \Delta V_{13}^n) \\
& + \frac{\partial}{\partial y} (-\Delta F^n + \Delta V_{21}^n + \Delta V_{22}^n + \Delta V_{23}^n) \\
& + \frac{\partial}{\partial z} (-\Delta G^n + \Delta V_{31}^n + \Delta V_{32}^n + \Delta V_{33}^n) \Bigg] \\
& + \frac{\Delta t}{1+\xi} \Bigg[\frac{\partial}{\partial x} (-E + V_{11} + V_{12} + V_{13})^n \\
& + \frac{\partial}{\partial y} (-F + V_{21} + V_{22} + V_{23})^n \\
& + \frac{\partial}{\partial z} (-G + V_{31} + V_{32} + V_{33})^n \Bigg] \\
& + \frac{\xi}{1+\xi} \Delta U^{n-1} \qquad \text{(B.3)}
\end{aligned}$$

where $\Delta E^n = E^{n+1} - E^n$, etc.

Note that ΔE^n, ΔF^n, ΔG^n and ΔV_{ij}^n are nonlinear functions of the conserved variables U. A <u>linear</u> equation with the same temporal accuracy as Eq. (B.3) can be obtained by expanding ΔE^n, ΔF^n, ΔG^n and ΔV_{ij}^n in a Taylor series, thus

$$E^{n+1} = E^n + \left(\frac{\partial E}{\partial U}\right)^n \Delta U^n$$

or

$$\Delta E^n = A^n \Delta U^n \qquad \text{(B.4a)}$$

Similarly

$$\Delta F^n = \left(\frac{\partial F}{\partial U}\right)^n \Delta U^n = B^n \Delta U^n \tag{B.4b}$$

$$\Delta G^n = \left(\frac{\partial G}{\partial U}\right)^n \Delta U^n = C^n \Delta U^n \tag{B.4c}$$

where A, B, and C are the linearization Jacobians.

Strictly speaking, the same procedure should be applied to the viscous terms. However, as pointed out in Ref. B.1, treating the spatial cross-derivative terms, i.e., ΔV_{ij} $(i \neq j)$, in this manner would lead to considerable difficulties in constructing an efficient spatially factored algorithm. Therefore, spatial cross-derivative terms will be evaluated explicitly (without loss of accuracy, Ref. 1), i.e.,

$$\Delta V_{ij}^n = \Delta V_{ij}^{n-1} \quad (i \neq j) \tag{B.5}$$

while the linearization is applied to the ΔV_{kk} $(k = 1, 2, 3)$. Remembering that $\Delta V_{kk} = f(U, U_{x_k})$,

$$\begin{aligned} \Delta V_{kk}^n &= \left(\frac{\partial V_{kk}}{\partial U}\right)^n \Delta U^n + \left(\frac{\partial V_{kk}}{\partial U_{x_k}}\right)^n \Delta U_{x_k}^n \\ &= P_{kk} \Delta U^n + R_{kk}^n \Delta U_{x_k}^n \end{aligned} \tag{B.6}$$

Application of the product differentiation rule shows that

$$R_{kk}^n \Delta U_{x_k}^n = \left(R_{kk} \Delta U\right)_{x_k}^n - \frac{\partial}{\partial x_k} R_{kk} \cdot \Delta U^n \tag{B.7}$$

and therefore from Eq. (B.6) and (B.7)

$$\Delta V_{kk}^{n} = P_{kk}\,\Delta U^{n} + (R_{kk}\,\Delta U)_{x_k}^{n} - R_{kk,x_k}\,\Delta U^{n}$$

$$= (P_{kk} - R_{kk,x_k})^{n}\,\Delta U^{n} + (R_{kk}\,\Delta U)_{x_k}^{n} \qquad \text{(B.8)}$$

where P_{kk} and R_{kk} are the linearization Jacobians as defined in Eq. (B.6). Evaluation of these Jacobians will show that for <u>constant</u> transport coefficients

$$P_{kk} - \frac{\partial}{\partial x_k} R_{kk} = 0 \qquad \text{(B.9)}$$

and thus

$$\Delta V_{kk}^{n} = \frac{\partial}{\partial x_k} (R_{kk}\,\Delta U)^{n} \qquad \text{(B.10)}$$

If the approximations outlined above are introduced into Eq. (B.3), we obtain

$$\begin{aligned}
\Delta U^{n} = \frac{\theta \Delta t}{1+\xi} \Bigg\{ & \frac{\partial}{\partial x}\left[-A^{n}\,\Delta U^{n} + \frac{\partial}{\partial x}(R_{11}\,\Delta U)^{n}\right] \\
& + \frac{\partial}{\partial y}\left[-B^{n}\,\Delta U^{n} + \frac{\partial}{\partial y}(R_{22}\,\Delta U)^{n}\right] \\
& + \frac{\partial}{\partial z}\left[-C^{n}\,\Delta U^{n} + \frac{\partial}{\partial z}(R_{33}\,\Delta U)^{n}\right]\Bigg\} \\
+ \frac{\theta \Delta t}{1+\xi} \Bigg[& \frac{\partial}{\partial x}(\Delta V_{12} + \Delta V_{13})^{n-1} \\
& + \frac{\partial}{\partial y}(\Delta V_{21} + \Delta V_{23})^{n-1} \\
& + \frac{\partial}{\partial z}(\Delta V_{31} + \Delta V_{32})^{n-1}\Bigg]
\end{aligned}$$

(Continued)

$$+ \frac{\Delta t}{1+\xi}\left[\frac{\partial}{\partial x}(-E + V_{11} + V_{12} + V_{13})^n\right.$$

$$+ \frac{\partial}{\partial y}(-F + V_{21} + V_{22} + V_{23})^n$$

$$\left. + \frac{\partial}{\partial z}(-G + V_{31} + V_{32} + V_{33})^n\right]$$

$$+ \frac{\xi}{1+\xi}\Delta U^{n-1} \tag{B.11}$$

Thus, for constant transport coefficients only the R_{kk} linearization Jacobians are needed in addition to the A, B and C Jacobians.

Expanding and rearranging Eq. (B.11), we obtain

$$\left\{I + \frac{\theta\Delta t}{1+\xi}\left[\frac{\partial}{\partial x}A^n - \frac{\partial^2}{\partial x^2}R^n_{11} + \frac{\partial}{\partial y}B^n - \frac{\partial^2}{\partial y^2}R^n_{22}\right.\right.$$

$$\left.\left. + \frac{\partial}{\partial z}C^n - \frac{\partial^2}{\partial z^2}R^n_{33}\right]\right\} * \Delta U^n$$

$$= \frac{\theta\Delta t}{1+\xi}\left[\frac{\partial}{\partial x}(\Delta V_{12} + \Delta V_{13})^{n-1} + \frac{\partial}{\partial y}(\Delta V_{21} + \Delta V_{23})^{n-1}\right.$$

$$\left. + \frac{\partial}{\partial z}(\Delta V_{31} + \Delta V_{32})^{n-1}\right]$$

$$+ \frac{\Delta t}{1+\xi}\left[\frac{\partial}{\partial x}(-E + V_{11} + V_{12} + V_{13})^n\right.$$

$$+ \frac{\partial}{\partial y}(-F + V_{21} + V_{22} + V_{23})^n$$

$$\left. + \frac{\partial}{\partial z}(-G + V_{31} + V_{32} + V_{33})^n\right]$$

$$+ \frac{\xi}{1+\xi}\Delta U^{n-1} \tag{B.12}$$

Note the special notation used in writing the left hand side (LHS) of Eq. (B.12) which really must be considered an "operator," operating on ΔU^n, and which is of the form

$$\mathrm{LHS}(12) = \left\{ I + a + b + c \right\} * \Delta U^n \tag{B.13}$$

This can be written in a spatially factored form

$$\begin{aligned} \mathrm{LHS}(12) &= \left\{ (I + a)\,(I + b)\,(I + c) \right\} * \Delta U^n \\ &= \left\{ (I + a + b + c) \right. \\ &\quad \left. + ab + ac + bc + abc \right\} * \Delta U^n \end{aligned} \tag{B.14}$$

if we note that ab, ac, bc, and abc all are at least an order of magnitude (in Δt) smaller than a, b, c. Thus, without loss of accuracy,

$$\begin{aligned} \mathrm{LHS}(12) &= \\ &= \left\{ \left[I + \frac{\theta \Delta t}{1+\xi} \left(\frac{\partial}{\partial x} A^n - \frac{\partial^2}{\partial x^2} R^n_{11} \right) \right] * \right. \\ &\quad \left[I + \frac{\theta \Delta t}{1+\xi} \left(\frac{\partial}{\partial y} B^n - \frac{\partial^2}{\partial x^2} R^n_{22} \right) \right] * \\ &\quad \left. \left[I + \frac{\theta \Delta t}{1+\xi} \left(\frac{\partial}{\partial z} C^n - \frac{\partial^2}{\partial z^2} R^n_{33} \right) \right] \right\} * \Delta U^n \end{aligned} \tag{B.15}$$

Following Beam and Warming (Ref. B-1), in practice Eq. (B.15) is implemented by the sequence

$$\left\{ I + \frac{\theta \Delta t}{1+\xi} \left(\frac{\partial}{\partial x} A^n - \frac{\partial^2}{\partial x^2} R^n_{11} \right) \right\} * \Delta U^{**} = \mathrm{RHS}(12) \tag{B.16a}$$

$$\left\{ I + \frac{\theta \Delta t}{1+\xi} \left(\frac{\partial}{\partial y} B^n - \frac{\partial^2}{\partial y^2} R^n_{22} \right) \right\} * \Delta U^{*} = \Delta U^{**} \tag{B.16b}$$

where RHS(12) means the right hand side of Eq. (B.12).

$$\left\{ I + \frac{\theta \Delta t}{1+\xi} \left(\frac{\partial}{\partial z} C^n - \frac{\partial^2}{\partial z^2} R_{33}^n \right) \right\} * \Delta U^n = \Delta U^* \tag{B.16c}$$

$$U^{n+1} = U^n + \Delta U^n \tag{B.16d}$$

The remainder of the analysis follows that of Ref. B-1.

B.3 SUMMARY OF EQUATIONS FOR UNSTEADY ALGORITHM

The vector of conserved variables, U, and the flux vectors of Eq. (B.1) are

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \equiv \begin{bmatrix} \rho \\ m \\ n \\ q \\ r \end{bmatrix}$$

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(\rho E + p) \end{bmatrix} = \begin{bmatrix} m \\ m^2/\rho + p \\ mn/\rho \\ mq/\rho \\ (m/\rho)\ (r + p) \end{bmatrix}$$

$$F = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(\rho E + p) \end{bmatrix} = \begin{bmatrix} n \\ mn/\rho \\ n^2/\rho + p \\ nq/\rho \\ (n/\rho)\ (r + p) \end{bmatrix}$$

$$G = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(\rho E + p) \end{bmatrix} = \begin{bmatrix} q \\ mq/\rho \\ nq/\rho \\ q^2/\rho + p \\ (q/\rho)\ (r + p) \end{bmatrix}$$

where the pressure is given by the equation of state

$$p = \rho(\gamma-1)\left(E - \frac{u^2 + v^2 + w^2}{2}\right)$$

$$= (\gamma-1)\left(r - \frac{m^2 + n^2 + q^2}{2\rho}\right)$$

The viscous flux vectors are

$$V_{11} = \begin{bmatrix} 0 \\ (2\mu + \lambda)\, u_x \\ \mu v_x \\ \mu w_x \\ (2\mu + \lambda)\, uu_x + \mu vv_x + \mu ww_x + kT_x \end{bmatrix}$$

$$V_{12} = \begin{bmatrix} 0 \\ \lambda v_y \\ \mu u_y \\ 0 \\ \lambda uv_y + \mu vu_y \end{bmatrix}$$

$$V_{13} = \begin{bmatrix} 0 \\ \lambda w_z \\ 0 \\ \mu u_z \\ \lambda uw_z + \mu wu_z \end{bmatrix}$$

$$V_{21} = \begin{bmatrix} 0 \\ \mu v_x \\ \lambda u_x \\ 0 \\ \mu uv_x + \lambda vu_x \end{bmatrix}$$

$$V_{22} = \begin{bmatrix} 0 \\ \mu u_y \\ (2\mu + \lambda)\, v_y \\ \mu w_y \\ \mu u u_y + (2\mu + \lambda)\, v v_y + \mu w w_y + k T_y \end{bmatrix}$$

$$V_{23} = \begin{bmatrix} 0 \\ 0 \\ \lambda w_z \\ \mu v_z \\ \lambda v w_z + \mu w v_z \end{bmatrix}$$

$$V_{31} = \begin{bmatrix} 0 \\ \mu w_x \\ 0 \\ \lambda u_x \\ \mu u w_x + \lambda w u_x \end{bmatrix}$$

$$V_{32} = \begin{bmatrix} 0 \\ 0 \\ \mu w_y \\ \lambda v_y \\ \mu v w_y + \lambda w v_y \end{bmatrix}$$

$$V_{33} = \begin{bmatrix} 0 \\ \mu u_z \\ \mu v_z \\ (2\mu + \lambda)\, w_z \\ \mu u u_z + \mu v v_z + (2\mu + \lambda)\, w w_z + k T_z \end{bmatrix}$$

where $u_x = \partial u/\partial x$, etc.

In order to write the viscous flux vectors in terms of the conserved variables, the temperature gradients must be expressed in terms of the conserved variables. It is easily shown that

$$k\frac{\partial T}{\partial \xi} = \mu\frac{\gamma}{\mathrm{Pr}}\cdot\frac{1}{\rho^3}\left\{\rho\left(\rho\frac{\partial r}{\partial \xi} - r\frac{\partial \rho}{\partial \xi}\right)\right.$$

$$- m\left(\rho\frac{\partial m}{\partial \xi} - m\frac{\partial \rho}{\partial \xi}\right) - n\left(\rho\frac{\partial n}{\partial \xi} - n\frac{\partial \rho}{\partial \xi}\right)$$

$$\left. - q\left(\rho\frac{\partial q}{\partial \xi} - q\frac{\partial \rho}{\partial \xi}\right)\right\}$$

Using this equation it can be shown that

$$V_{11} = \begin{bmatrix} 0 \\ (2\mu+\lambda)\,\rho^{-2}\,(\rho m_x - m\rho_x) \\ \mu\rho^{-2}\,(\rho n_x - n\rho_x) \\ \mu\rho^{-2}\,(\rho q_x - q\rho_x) \\ \mu\rho^{-3}\left\{(2 + \frac{\lambda}{\mu} - \frac{\gamma}{\mathrm{Pr}})\,m\,(\rho m_x - m\rho_x)\right. \\ \quad + (1 - \frac{\gamma}{\mathrm{Pr}})\left[n(\rho n_x - n\rho_x) + q\,(\rho q_x - q\rho_x)\right] \\ \quad \left. + \frac{\gamma}{\mathrm{Pr}}\,\rho(\rho r_x - r\rho_x)\right\} \end{bmatrix}$$

$$V_{12} = \begin{bmatrix} 0 \\ \lambda\rho^{-2}\,(\rho n_y - n\rho_y) \\ \mu\rho^{-2}\,(\rho m_y - m\rho_y) \\ 0 \\ \lambda\rho^{-3}\,m(\rho n_y - n\rho_y) + \mu\rho^{-3}\,n(\rho m_y - m\rho_y) \end{bmatrix}$$

$$V_{13} = \begin{bmatrix} 0 \\ \lambda\rho^{-2}\,(\rho q_z - q\rho_z) \\ 0 \\ \mu\rho^{-2}\,(\rho m_z - m\rho_z) \\ \lambda\rho^{-3}\,m(\rho q_z - q\rho_z) + \mu\rho^{-3}\,q(\rho m_z - m\rho_z) \end{bmatrix}$$

$$V_{21} = \begin{bmatrix} 0 \\ \mu\rho^{-2}\,(\rho n_x - n\rho_x) \\ \lambda\rho^{-2}\,(\rho m_x - m\rho_x) \\ 0 \\ \mu\rho^{-3}\,m(\rho n_x - n\rho_x) + \lambda\rho^{-3}\,n(\rho m_x - m\rho_x) \end{bmatrix}$$

$$V_{22} = \begin{bmatrix} 0 \\ \mu\rho^{-2}\,(\rho m_y - m\rho_y) \\ (2\mu + \lambda)\,\rho^{-2}\,(\rho n_y - n\rho_y) \\ \mu\rho^{-2}\,(\rho q_y - q\rho_y) \\ \mu\rho^{-3}\left\{(2 + \frac{\lambda}{\mu} - \frac{\gamma}{Pr})\,n(\rho n_y - n\rho_y) \right. \\ \quad + (1 - \frac{\gamma}{Pr})\left[m(\rho m_y - m\rho_y) + q(\rho q_y - q\rho_y)\right] \\ \quad \left. + \frac{\gamma}{Pr}\,\rho\,(\rho r_y - r\rho_y)\right\} \end{bmatrix}$$

$$V_{23} = \begin{bmatrix} 0 \\ 0 \\ \lambda\rho^{-2}\,(\rho q_z - q\rho_z) \\ \mu\rho^{-2}\,(\rho n_z - n\rho_z) \\ \lambda\rho^{-3}\,n(\rho q_z - q\rho_z) + \mu\rho^{-3}\,q(\rho n_z - n\rho_z) \end{bmatrix}$$

$$V_{31} = \begin{bmatrix} 0 \\ \mu\rho^{-2}(\rho q_x - q\rho_x) \\ 0 \\ \lambda\rho^{-2}(\rho m_x - m\rho_x) \\ \mu\rho^{-3} m(\rho q_x - q\rho_x) + \lambda\rho^{-3} q(\rho m_x - m\rho_x) \end{bmatrix}$$

$$V_{32} = \begin{bmatrix} 0 \\ 0 \\ \mu\rho^{-2}(\rho q_y - q\rho_y) \\ \lambda\rho^{-2}(\rho n_y - n\rho_y) \\ \mu\rho^{-3} n(\rho q_y - q\rho_y) + \lambda\rho^{-3} q(\rho n_y - n\rho_y) \end{bmatrix}$$

$$V_{33} = \begin{bmatrix} 0 \\ \mu\rho^{-2}(\rho m_z - m\rho_z) \\ \mu\rho^{-2}(\rho n_z - n\rho_z) \\ (2\mu+\lambda)\,\rho^{-2}(\rho q_z - q\rho_z) \\ \mu\rho^{-3}\left\{(2 + \frac{\lambda}{\mu} - \frac{\gamma}{Pr})\, q(\rho q_z - q\rho_z) \right. \\ \quad + (1 - \frac{\gamma}{Pr})\left[m(\rho m_z - m\rho_z) + n(\rho n_z - n\rho_z)\right] \\ \left. \quad + \frac{\gamma}{Pr}\,\rho(\rho r_z - r\rho_z)\right\} \end{bmatrix}$$

The linearization Jacobians A, B and C are:

$$A = \frac{\partial E}{\partial U} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{\gamma-1}{2}(u^2+v^2+w^2)-u^2 & -(\gamma-3)u & -(\gamma-1)v & -(\gamma-1)w & (\gamma-1) \\ -uv & v & u & 0 & 0 \\ -uw & w & 0 & u & 0 \\ u\left[\gamma E-\frac{2}{\rho}(\rho E+p)\right] & \frac{1}{\rho}(\rho E+p)-(\gamma-1)u^2 & -(\gamma-1)uv & -(\gamma-1)uw & \gamma u \end{bmatrix}$$

$$B = \frac{\partial F}{\partial U} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -uv & v & u & 0 & 0 \\ \frac{\gamma-1}{2}(u^2+v^2+w^2)-v^2 & -(\gamma-1)u & -(\gamma-3)v & -(\gamma-1)w & (\gamma-1) \\ -vw & 0 & w & v & 0 \\ v\left[\gamma E-\frac{2}{\rho}(\rho E+p)\right] & -(\gamma-1)uv & \frac{1}{\rho}(\rho E+p)-(\gamma-1)v^2 & -(\gamma-1)vw & \gamma v \end{bmatrix}$$

$$C = \frac{\partial G}{\partial U} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -uw & w & 0 & u & 0 \\ -vw & 0 & w & v & 0 \\ \frac{\gamma-1}{2}(u^2+v^2+w^2)-w^2 & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-3)w & (\gamma-1) \\ w\left[\gamma E-\frac{2}{\rho}(\rho E+p)\right] & -(\gamma-1)uw & -(\gamma-1)vw & \frac{1}{\rho}(\rho E+p)-(\gamma-1)w^2 & \gamma w \end{bmatrix}$$

The linearization Jacobians R_{11}, R_{22}, R_{33} are:

$$R_{11} = \frac{\partial V_{11}}{\partial U_x} = -\frac{\mu}{\rho}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ (2+\frac{\lambda}{\mu})\,u & -(2+\frac{\lambda}{\mu}) & 0 & 0 & 0 \\ v & 0 & -1 & 0 & 0 \\ w & 0 & 0 & -1 & 0 \\ \begin{array}{l}(1-\frac{\gamma}{Pr})\,(u^2+v^2+w^2) \\ +\frac{\gamma}{Pr}\,E + (1+\frac{\lambda}{\mu})\,u^2\end{array} & -(2+\frac{\lambda}{\mu}-\frac{\gamma}{Pr})\,u & -(1-\frac{\gamma}{Pr})\,v & -(1-\frac{\gamma}{Pr})\,w & -\frac{\gamma}{Pr} \end{bmatrix}$$

$$R_{22} = \frac{\partial V_{22}}{\partial U_y} = -\frac{\mu}{\rho}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ u & -1 & 0 & 0 & 0 \\ (2+\frac{\lambda}{\mu})\,v & 0 & -(2+\frac{\lambda}{\mu}) & 0 & 0 \\ w & 0 & 0 & -1 & 0 \\ \begin{array}{l}(1-\frac{\gamma}{Pr})\,(u^2+v^2+w^2) \\ +\frac{\gamma}{Pr}\,E + (1+\frac{\lambda}{\mu})\,v^2\end{array} & -(1-\frac{\gamma}{Pr})\,u & -(2+\frac{\lambda}{\mu}-\frac{\gamma}{Pr})\,v & -(1-\frac{\gamma}{Pr})\,w & -\frac{\gamma}{Pr} \end{bmatrix}$$

$$R_{33} = \frac{\partial V_{33}}{\partial U_z} = -\frac{\mu}{\rho}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ u & -1 & 0 & 0 & 0 \\ v & 0 & -1 & 0 & 0 \\ (2+\frac{\lambda}{\mu})\,w & 0 & 0 & -(2+\frac{\lambda}{\mu}) & 0 \\ \begin{array}{l}(1-\frac{\gamma}{Pr})\,(u^2+v^2+w^2) \\ +\frac{\gamma}{Pr}\,E + (1+\frac{\lambda}{\mu})\,w^2\end{array} & -(1-\frac{\gamma}{Pr})\,u & -(1-\frac{\gamma}{Pr})\,v & -(2+\frac{\lambda}{\mu}-\frac{\gamma}{Pr})\,w & -\frac{\gamma}{Pr} \end{bmatrix}$$

B.4 THREE-DIMENSIONAL STEADY PARABOLIZED ALGORITHM

A flow model which uses a spatial forward marching procedure in the principal direction of flow to obtain a solution cannot tolerate the upstream propagation of any flow phenomena. Such a model is obtained by deleting those viscous terms from the governing equations which contain gradients in the marching direction, and the resulting set of equations is termed "parabolized."

The three-dimensional, steady state compressible conservation equations for parabolic flow (in the x-direction) are

$$\frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = \frac{\partial}{\partial y}\left[V_{22}(E, E_y) + V_{23}(E, E_z)\right] + \frac{\partial}{\partial z}\left[V_{32}(E, E_y) + V_{33}(E, E_z)\right] \qquad (B.17)$$

where E is the vector of conserved variables, and F, G and V_{ij} are flux vectors.

In complete analogy to the treatment of the unsteady problem, a generalized single-step spatial finite difference scheme for advancing the solution of Eq. (B.17) can be written as

$$\Delta E^n = \frac{\theta \Delta x}{1+\xi} \frac{\partial}{\partial x} \Delta E^n + \frac{\Delta x}{1+\xi} \frac{\partial E^n}{\partial x} + \frac{\xi}{1+\xi} \Delta E^{n-1} \qquad (B.18)$$

where $E^n = E(n\Delta x, y, z)$ and $\Delta E^n = E^{n+1} - E^n$ (terms of order Δx^2 and Δx^3 have been negelected for simplicity).

Solving Eq. (B.17) for $\partial E/\partial x$ and substituting into Eq. (B.18) yields

$$\Delta E^n = \frac{\theta\Delta x}{1+\xi}\left[\frac{\partial}{\partial y}(-\Delta F + \Delta V_{22} + \Delta V_{23})^n + \frac{\partial}{\partial z}(-\Delta G + \Delta V_{32} + \Delta V_{33})^n\right] + \frac{\Delta x}{1+\xi}\left[\frac{\partial}{\partial y}(-F + V_{22} + V_{23})^n + \frac{\partial}{\partial z}(-G + V_{32} + V_{33})^n\right] + \frac{\xi}{1+\xi}\Delta E^{n-1} \tag{B.19}$$

Again, ΔF^n, ΔG^n and ΔV_{ij}^n are nonlinear functions of the conserved variables E. A linear equation with the same spatial accuracy as Eq. (B.19) can be obtained by expanding ΔF^n, ΔG^n and ΔV_{kk}^n in a Taylor series while treating ΔV_{ij}^n $(i \neq j)$ explicitly. Accordingly,

$$\Delta F^n = \left(\frac{\partial F}{\partial E}\right)^n \Delta E^n = A^n \Delta E^n \tag{B.20a}$$

$$\Delta G^n = \left(\frac{\partial G}{\partial E}\right)^n \Delta E^n = B^n \Delta E^n \tag{B.20b}$$

where A and B are linearization Jacobians.

While assuming that

$$\Delta V_{ij}^n = \Delta V_{ij}^{n-1} \quad (i \neq j) \tag{B.21}$$

we can write

$$\Delta V_{kk}^n = \left(\frac{\partial V_{kk}}{\partial E}\right)^n \Delta E^n + \left(\frac{\partial V_{kk}}{\partial E_{x_k}}\right)^n \Delta E_{x_k}^n = P_{kk}\Delta E^n + R_{kk}^n \Delta E_{x_k}^n \tag{B.22}$$

Since

$$R^n_{kk} \frac{\partial}{\partial x_k} \Delta E^n = \frac{\partial}{\partial x_k} (R^n_{kk} \Delta E^n) - \frac{\partial}{\partial x_k} R^n_{kk} \Delta E^n$$

we can rewrite

$$\Delta V^n_{kk} = (P^n_{kk} - \frac{\partial}{\partial x_k} R^n_{kk}) \Delta E^n + \frac{\partial}{\partial x_k} (R^n_{kk} \Delta E^n) \qquad (B.23)$$

where P_{kk} and R_{kk} are the linearization Jacobians defined in Eq. (B.22).

Using the viscous flux vectors V_{22} and V_{33}, it can be shown again that, for constant transport coefficients,

$$P^n_{kk} - \frac{\partial}{\partial x_k} R^n_{kk} = 0 \qquad (B.24)$$

which allows us to simplify Eq. (23) to

$$\Delta V^n_{kk} = \frac{\partial}{\partial x_k} (R^n_{kk} \Delta E^n) \qquad (B.25)$$

Substituting Eqs.

$$\begin{aligned}
\Delta E^n = {} & \frac{\theta \Delta x}{1+\xi} \left\{ \frac{\partial}{\partial y} \left[- A^n \Delta E^n + \frac{\partial}{\partial y} (R^n_{22} \Delta E^n) \right] \right. \\
& \left. + \frac{\partial}{\partial z} \left[- B^n \Delta E^n + \frac{\partial}{\partial z} (R^n_{33} \Delta E^n) \right] \right\} \\
& + \frac{\theta \Delta x}{1+\xi} \left[\frac{\partial}{\partial y} (\Delta V_{23})^{n-1} + \frac{\partial}{\partial z} (\Delta V_{32})^{n-1} \right] \\
& + \frac{\Delta x}{1+\xi} \left[\frac{\partial}{\partial y} (- F + V_{22} + V_{23})^n \right. \\
& \left. + \frac{\partial}{\partial z} (- G + V_{32} + V_{33})^n \right] \\
& + \frac{\xi}{1+\xi} \Delta E^{n-1} \qquad (B.26)
\end{aligned}$$

From Eq. (B.26) it is concluded that in addition to the A and B Jacobians we only need R_{kk} $(k = 2,3)$ if constant transport coefficients are assumed.

Expanding and rearranging Eq. (B.26) to combine all terms containing ΔE^n, we obtain

$$\left\{I + \frac{\theta\Delta x}{1+\xi}\left[\frac{\partial}{\partial y} A^n - \frac{\partial^2}{\partial y^2} R_{22}^n + \frac{\partial}{\partial z} B^n - \frac{\partial^2}{\partial z^2} R_{33}^n\right]\right\} * \Delta E^n$$

$$= \frac{\theta\Delta x}{1+\xi}\left[\frac{\partial}{\partial y} \Delta V_{23}^{n-1} + \frac{\partial}{\partial z} \Delta V_{32}^{n-1}\right]$$

$$+ \frac{\Delta x}{1+\xi}\left[\frac{\partial}{\partial y}(-F + V_{22} + V_{23})^n + \frac{\partial}{\partial z}(-G + V_{32} + V_{33})^n\right]$$

$$+ \frac{\xi}{1+\xi} \Delta E^{n-1} \tag{B.27}$$

In analogous fashion to the unsteady formulation and without loss of accuracy, we can write

$$\text{LHS}(27) = \left\{\left[I + \frac{\theta\Delta x}{1+\xi}\left(\frac{\partial}{\partial y} A^n - \frac{\partial^2}{\partial y^2} R_{22}^n\right)\right] * \right.$$

$$\left.\left[I + \frac{\theta\Delta x}{1+\xi}\left(\frac{\partial}{\partial z} B^n - \frac{\partial^2}{\partial z^2} R_{33}^n\right)\right]\right\} * \Delta E^n \tag{B.28}$$

where LHS(27) means the left hand side of Eq. (B.27), which in practice is implemented by the sequence

$$\left[I + \frac{\theta\Delta x}{1+\xi}\left(\frac{\partial}{\partial y} A^n - \frac{\partial^2}{\partial y^2} R_{22}^n\right)\right] * \Delta E^* = \text{RHS}(27) \tag{B.29a}$$

$$\left[I + \frac{\theta\Delta x}{1+\xi}\left(\frac{\partial}{\partial z} B^n - \frac{\partial^2}{\partial z^2} R_{33}^n\right)\right] * \Delta E^n = \Delta E^* \tag{B.29b}$$

and, finally

$$E^{n+1} = E^n + \Delta E^n \tag{B.29c}$$

where RHS(27) means the right hand side of Eq. (B.27).

B.5 SUMMARY OF EQUATIONS FOR STEADY STATE ALGORITHM

For steady flow, the vector of conserved variables, E, and the flux vectors of Eq.(B.17) are

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(\rho E + p) \end{bmatrix} \equiv \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \end{bmatrix}$$

$$F = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(\rho E + p) \end{bmatrix} = \begin{bmatrix} E_3/u \\ E_3 \\ E_2 + E_3^2/(E_1 u) - E_1 u \\ E_3\, E_4/(E_1 u) \\ E_3\, E_5/(E_1 u) \end{bmatrix}$$

$$G = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(\rho E + p) \end{bmatrix} = \begin{bmatrix} E_4/u \\ E_4 \\ E_3\, E_4/(E_1 u) \\ E_2 + E_4^2/(E_1 u) - E_1 u \\ E_4\, E_5/(E_1 u) \end{bmatrix}$$

where $u = f(E_1, E_2, E_3, E_4, E_5)$ as obtained by decoding the E vector for the primitive variables. It should be noted that although the flux vectors for the steady case are the same as for the unsteady case, their form in terms of the conserved variables differs from that in the unsteady case. It is assumed that decoding the E vector was accomplished by obtaining u as a function of the E vector components, i.e.,

$$u = \frac{\gamma}{\gamma+1}\left\{\frac{E_2}{E_1} \underline{(+)} \sqrt{\left(\frac{E_2}{E_1}\right)^2 - \frac{\gamma^2-1}{\gamma^2}\left[2\frac{E_5}{E_1} - \left(\frac{E_3}{E_1}\right)^2 - \left(\frac{E_4}{E_1}\right)^2\right]}\right\}$$

and

$$v = E_3/E_1$$

$$w = E_4/E_1$$

$$\rho = E_1/u$$

$$p = E_2 - E_1 u$$

$$E = (E_5 - up)/E_1$$

It can be shown that in the decode procedure the (+) and the (-) sign apply to supersonic and to subsonic flow, respectively. As long as the flow is strictly supersonic or subsonic, choice of the sign should be no problem. It is obviously a problem for transonic flows, boundary layers and supersonic flows with imbedded subsonic pockets.

In terms of the conserved variables, the heat conduction term becomes

$$k\frac{\partial T}{\partial \xi} = \frac{\mu}{Pr}\frac{\partial}{\partial \xi}\left(\frac{E_5}{E_1}\right) - \frac{\mu}{Pr}\frac{\partial}{\partial \xi}\left\{\frac{1}{2}\left[u^2 + \left(\frac{E_3}{E_1}\right)^2 + \left(\frac{E_4}{E_1}\right)^2\right]\right\}$$

Using this equation the viscous flux vectors V_{ij} in Eq.(B.17) can be written as shown on the following page.

$$V_{22} = \begin{bmatrix} 0 \\ \mu \sum_{i=1}^{5} \left(\frac{\partial u}{\partial E_i}\right) \frac{\partial E_i}{\partial y} \\ (2\mu + \lambda)\ E_1^{-2} \left(E_1 \frac{\partial E_3}{\partial y} - E_3 \frac{\partial E_1}{\partial y}\right) \\ \mu E_1^{-2} \left(E_1 \frac{\partial E_4}{\partial y} - E_4 \frac{\partial E_1}{\partial y}\right) \\ \mu\,(1 - \frac{1}{Pr})\ u \sum_{i=1}^{5} \left(\frac{\partial u}{\partial E_i}\right) \frac{\partial E_i}{\partial y} \\ \quad + \mu\,(2 + \frac{\lambda}{\mu} - \frac{1}{Pr})\ E_1^{-3}\ E_3 \left(E_1 \frac{\partial E_3}{\partial y} - E_3 \frac{\partial E_1}{\partial y}\right) \\ \quad + \mu\,(1 - \frac{1}{Pr})\ E_1^{-3}\ E_4 \left(E_1 \frac{\partial E_4}{\partial y} - E_4 \frac{\partial E_1}{\partial y}\right) \\ \quad + \frac{\mu}{Pr}\ E_1^{-2} \left(E_1 \frac{\partial E_5}{\partial y} - E_5 \frac{\partial E_1}{\partial y}\right) \end{bmatrix}$$

and

$$V_{23} = \begin{bmatrix} 0 \\ 0 \\ \lambda\ E_1^{-2} \left(E_1 \frac{\partial E_4}{\partial z} - E_4 \frac{\partial E_1}{\partial z}\right) \\ \mu E_1^{-2} \left(E_1 \frac{\partial E_3}{\partial z} - E_3 \frac{\partial E_1}{\partial z}\right) \\ \mu E_1^{-3}\ E_4 \left(E_1 \frac{\partial E_3}{\partial z} - E_3 \frac{\partial E_1}{\partial z}\right) \\ \quad + \lambda\ E_1^{-3}\ E_3 \left(E_1 \frac{\partial E_4}{\partial z} - E_4 \frac{\partial E_1}{\partial z}\right) \end{bmatrix}$$

$$
V_{32} \quad \begin{bmatrix}
0 \\
0 \\
\mu E_1^{-2} \left(E_1 \dfrac{\partial E_4}{\partial y} - E_4 \dfrac{\partial E_1}{\partial y} \right) \\
\lambda E_1^{-2} \left(E_1 \dfrac{\partial E_3}{\partial y} - E_3 \dfrac{\partial E_1}{\partial y} \right) \\
\mu E_1^{-3} E_3 \left(E_1 \dfrac{\partial E_4}{\partial y} - E_4 \dfrac{\partial E_1}{\partial y} \right) \\
\quad + \lambda E_1^{-3} E_4 \left(E_1 \dfrac{\partial E_3}{\partial y} - E_3 \dfrac{\partial E_1}{\partial y} \right)
\end{bmatrix}
$$

and

$$
V_{33} = \begin{bmatrix}
0 \\
\mu \displaystyle\sum_{i=1}^{5} \left(\frac{\partial u}{\partial E_i} \right) \frac{\partial E_i}{\partial z} \\
\mu E_1^{-2} \left(E_1 \dfrac{\partial E_3}{\partial z} - E_3 \dfrac{\partial E_1}{\partial z} \right) \\
(2\mu + \lambda) E_1^{-2} \left(E_1 \dfrac{\partial E_4}{\partial z} - \dfrac{\partial E_1}{\partial z} \right) \\
\mu \left(1 - \dfrac{1}{Pr}\right) u \displaystyle\sum_{i=1}^{5} \left(\frac{\partial u}{\partial E_i} \right) \frac{\partial E_i}{\partial z} \\
\quad + \mu \left(1 - \dfrac{1}{Pr}\right) E_1^{-3} E_3 \left(E_1 \dfrac{\partial E_3}{\partial z} - E_3 \dfrac{\partial E_1}{\partial z} \right) \\
\quad + \mu \left(2 + \dfrac{\lambda}{\mu} - \dfrac{1}{Pr}\right) E_1^{-3} E_4 \left(E_1 \dfrac{\partial E_4}{\partial z} - E_4 \dfrac{\partial E_1}{\partial z} \right) \\
\quad + \dfrac{\mu}{Pr} E_1^{-2} \left(E_1 \dfrac{\partial E_5}{\partial z} - E_5 \dfrac{\partial E_1}{\partial z} \right)
\end{bmatrix}
$$

The linearization Jacobians A and B are obtained by differentiating the flux vectors F and G with respect to the conserved variables. The result is

$$A = A'_{ij} - A''_{ij} \left(\frac{\partial u}{\partial E_j}\right)$$

$$B = B'_{ij} - B'_{ij} \left(\frac{\partial u}{\partial E_j}\right)$$

where $(\partial u/\partial E_j)$ is a column vector obtained by differentation.

$$A'_{ij} = \frac{1}{u} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & u & 0 & 0 \\ -(u^2 + v^2) & u & 2v & 0 & 0 \\ -vw & 0 & w & v & 0 \\ -v(E + \frac{p}{\rho}) & 0 & (E + \frac{p}{\rho}) & 0 & v \end{bmatrix}$$

$$A''_{ij} = \rho \frac{v}{u} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{u^2 + v^2}{v} & \frac{u^2 + v^2}{v} & \frac{u^2 + v^2}{v} & \frac{u^2 + v^2}{v} & \frac{u^2 + v^2}{v} \\ w & w & w & w & w \\ (E + \frac{p}{\rho}) & (E + \frac{p}{\rho}) & (E + \frac{p}{\rho}) & (E + \frac{p}{\rho}) & (E + \frac{p}{\rho}) \end{bmatrix}$$

and

$$B'_{ij} = \frac{1}{u}\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & u & 0 \\ -vw & 0 & w & v & 0 \\ -(u^2+w^2) & u & 0 & 2w & 0 \\ -w(E+\frac{p}{\rho}) & 0 & 0 & (E+\frac{p}{\rho}) & w \end{bmatrix}$$

$$B''_{ij} = \rho\frac{w}{u}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ v & v & v & v & v \\ \frac{u^2+w^2}{w} & \frac{u^2+w^2}{w} & \frac{u^2+w^2}{w} & \frac{u^2+w^2}{w} & \frac{u^2+w^2}{w} \\ (E+\frac{p}{\rho}) & (E+\frac{p}{\rho}) & (E+\frac{p}{\rho}) & (E+\frac{p}{\rho}) & (E+\frac{p}{\rho}) \end{bmatrix}$$

The components of the $(\partial u/\partial E)$ column vector are given by

$$\frac{\partial u}{\partial E_1} = \frac{\gamma}{(\gamma+1)E_1}\left\{-\frac{E_2}{E_1} (\pm)\,\Omega\left[\frac{\gamma^2-1}{\gamma^2}\left(\frac{E_5}{E_1} - \left(\frac{E_3}{E_1}\right)^2 - \left(\frac{E_4}{E_1}\right)^2\right) - \left(\frac{E_2}{E_1}\right)^2\right]\right\}$$

$$\frac{\partial u}{\partial E_2} = \frac{\gamma}{(\gamma+1)E_1}\left(1\ (\pm)\,\Omega\frac{E_2}{E_1}\right)$$

$$\frac{\partial u}{\partial E_3} = (\pm)\,\frac{(\gamma-1)\Omega}{\gamma E_1}\left(\frac{E_3}{E_1}\right)$$

$$\frac{\partial u}{\partial E_4} = (\pm)\,\frac{(\gamma-1)\Omega}{\gamma E_1}\left(\frac{E_4}{E_1}\right)$$

$$\frac{\partial u}{\partial E_5} = (\mp)\,\frac{(\gamma-1)\Omega}{\gamma E_1}$$

where

$$\Omega \equiv \left\{\left(\frac{E_2}{E_1}\right)^2 - \frac{\gamma^2-1}{\gamma^2}\left[2\,\frac{E_5}{E_1} - \left(\frac{E_3}{E_1}\right)^2 - \left(\frac{E_4}{E_1}\right)^2\right]\right\}^{-1/2}$$

The linearization Jacobians R_{kk} $(k = 2, 3)$ are obtained by differentiating the viscous flux vectors V_{kk} with respect to the spatial derivatives of the conserved variables E_i $(i = 1, 5)$. The result is:

$$R_{22} = R_{22}^1 + Q\left(\frac{\partial u}{\partial E}\right)$$

$$R_{33} = R_{33}^1 + Q\left(\frac{\partial u}{\partial E}\right)$$

$$R_{22}^1 = \frac{\mu}{\rho u}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -(2+\frac{\lambda}{\mu})\,v & 0 & (2+\frac{\lambda}{\mu}) & 0 & 0 \\ -w & 0 & 0 & 1 & 0 \\ \begin{array}{c} -(1+\frac{\lambda}{\mu})\,v^2 \\ -(1-\frac{1}{Pr})\,(v^2+w^2) \\ -\frac{1}{Pr}\,(E+\frac{p}{\rho}) \end{array} & 0 & (2+\frac{\lambda}{\mu}-\frac{1}{Pr})\,v & (1-\frac{1}{Pr})\,w & \frac{1}{Pr} \end{bmatrix}$$

$$R^{1}_{33} = \frac{\mu}{\rho u}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -v & 0 & 1 & 0 & 0 \\ -\left(2+\frac{\lambda}{\mu}\right) w & 0 & 0 & \left(2+\frac{\lambda}{\mu}\right) & 0 \\ \begin{array}{c} -\left(1+\frac{\lambda}{\mu}\right) w^{2} \\ -\left(1-\frac{1}{Pr}\right)\left(v^{2}+w^{2}\right) \\ -\frac{1}{Pr}\left(E+\frac{p}{\rho}\right) \end{array} & 0 & \left(1-\frac{1}{Pr}\right) v & \left(2+\frac{\lambda}{\mu}-\frac{1}{Pr}\right) w & \frac{1}{Pr} \end{bmatrix}$$

$$Q = \mu\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \left(1-\frac{1}{Pr}\right) u & \left(1-\frac{1}{Pr}\right) u & \left(1-\frac{1}{Pr}\right) u & \left(1-\frac{1}{Pr}\right) & \left(1-\frac{1}{Pr}\right) u \end{bmatrix}$$

B.6 REFERENCES

B-1. Beam, Richard M., and B. F. Warming, "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Paper No. 77-645 (1977).

Appendix C

MULTILINEAR INTERPOLANTS FOR GIM MARCHING METHODS

by

John F. Stalnaker

Appendix C

C.1 INTRODUCTION

The following is a study to determine the nature of the multilinear weight functions which will generate certain implicit, spatial marching finite difference schemes within the GIM framework. The derivations are performed using rectangular two- and three-dimensional grids for simplicity of understanding. In these grids the local and Cartesian coordinates coincide; however, it should be realized that this is not always, perhaps seldom, the case and that the finite difference scheme generated by the GIM code is in terms of the local coordinates.

With the above caveat aside let us proceed by setting in one place the notation to be used herein:

S_α	shape function for element point α
W_β	weight function for element point β
$D^{(k)}_{\alpha\beta}$	element difference operators for η_k direction
f, g, h	unique components of $D^{(1)}$, $D^{(2)}$, and $D^{(3)}$, respectively
E, F, G	vectors of conserved variables
ϵ, θ, ϕ	Beam-Warming marching parameters
a, b	finite difference parameters in the η_2 and η_3 directions, respectively
Δ_J	determinant of the Jacobian of transformation.

Subscripts

i, j, k	assembled grid point indices for η_1, η_2, and η_3, respectively.
	$= 1, \ldots, 9$ 2-D
	$= 1, \ldots, 27$ 3-D
α, β	element point indices $= 1, \ldots, 4$ 2-D; $= 1, \ldots 8$, 3-D

In all the following η_1 (and x in the case of the rectilinear grid) is assumed to be the marching direction.

C.2 TWO-DIMENSIONAL BILINEAR WEIGHT FUNCTIONS

For the two-dimensional case the shape functions are assumed to be the same bilinear shape functions now in the GIM code (Ref. C-1):

$$S_\alpha = \begin{cases} (1 - \eta_1)(1 - \eta_2) \\ \eta_1 (1 - \eta_2) \\ \eta_1 \eta_2 \\ (1 - \eta_1) \eta_2 \end{cases} \tag{C.1}$$

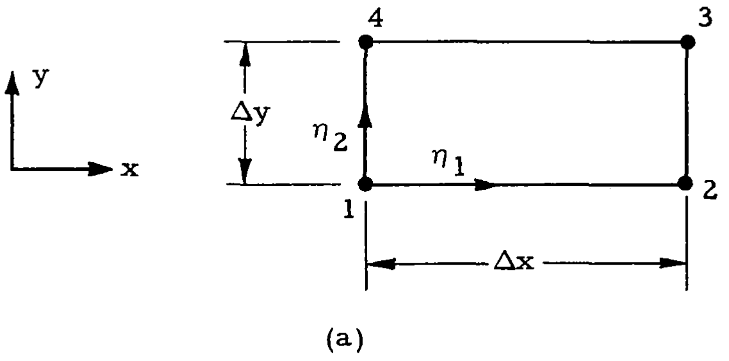
The weight functions are assumed to have the form

$$\Delta_J \bullet W_\beta = C_{\beta 0} - C_{\beta 1} \eta_1 - C_{\beta 2} \eta_2 + C_{\beta 3} \eta_1 \eta_2 \tag{C.2}$$

where the $C_{\beta i}$ are to be determined. The nodal numbering system for the individual elements and the nine node box are shown in Fig. C-1.

The two-dimensional element difference operators are

$$D^{(1)}_{\beta\alpha} = \int_0^1 d\eta_1 \int_0^1 d\eta_2 \, W_\beta \left[\frac{\partial S_\alpha}{\partial \eta_1} \frac{\partial y}{\partial \eta_2} - \frac{\partial S_\alpha}{\partial \eta_2} \frac{\partial y}{\partial \eta_1} \right] \tag{C.3}$$



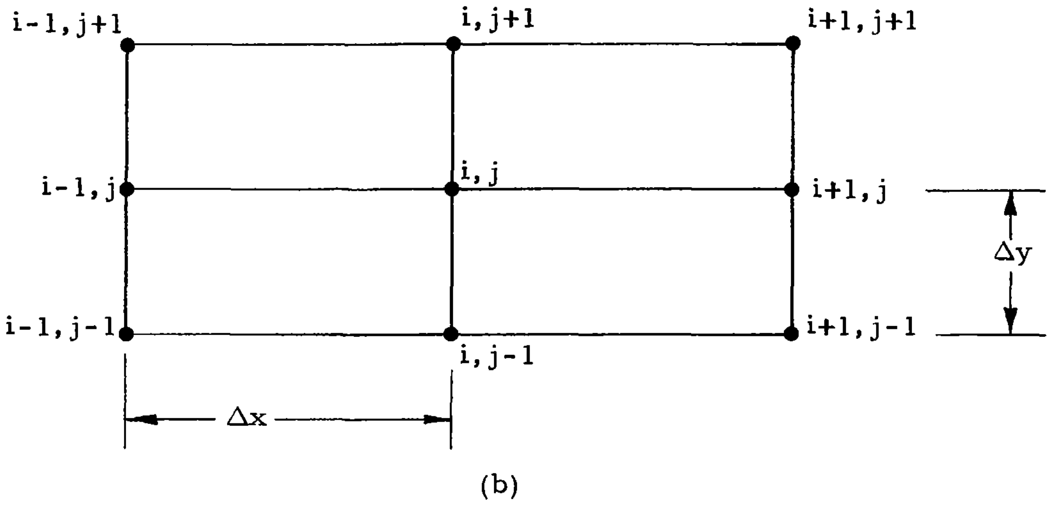


Fig. C-1 - Two-Dimensional Nodal Numbering System
(a) Element System; (b) Nine-Node Rectangle

$$D_{\beta\alpha}^{(2)} = \int_0^1 d\eta_1 \int_0^1 d\eta_2 \, W_\beta \left[\frac{\partial S_\alpha}{\partial \eta_2} \frac{\partial x}{\partial \eta_1} - \frac{\partial S_\alpha}{\partial \eta_1} \frac{\partial x}{\partial \eta_2} \right] \qquad \begin{matrix}\text{(C.3)}\\ \text{(Concl'd)}\end{matrix}$$

Noting that for the rectilinear element

$$\frac{\partial x}{\partial \eta_1} = \Delta x, \quad \frac{\partial y}{\partial \eta_2} = \Delta y, \quad \frac{\partial x}{\partial \eta_2} = \frac{\partial y}{\partial \eta_1} = 0, \qquad \Delta_J = \Delta x \, \Delta y$$

Substituting Eqs. (C.1) and (C.2) into (C.3) results in the following element difference operators

$$\frac{f_{\beta 1}}{\Delta x} \equiv D_{\beta 2}^{(1)} = -D_{\beta 1}^{(1)} = \frac{1}{12 \, \Delta x} \left[6 \, C_{\beta 0} - 3 \, C_{\beta 1} - 2 \, C_{\beta 2} + C_{\beta 3} \right]$$

$$\frac{f_{\beta 2}}{\Delta x} \equiv D_{\beta 3}^{(1)} = -D_{\beta 4}^{(1)} = \frac{1}{12 \, \Delta x} \left[6 \, C_{\beta 0} - 3 \, C_{\beta 1} - 4 \, C_{\beta 2} + 2 \, C_{\beta 3} \right]$$

(C.4)

$$\frac{g_{\beta 1}}{\Delta y} \equiv D_{\beta 4}^{(2)} = -D_{\beta 1}^{(2)} = \frac{1}{12 \, \Delta y} \left[6 \, C_{\beta 0} - 2 \, C_{\beta 1} - 3 \, C_{\beta 2} + C_{\beta 3} \right]$$

$$\frac{g_{\beta 2}}{\Delta y} \equiv D_{\beta 3}^{(2)} = -D_{\beta 2}^{(2)} = \frac{1}{12 \, \Delta y} \left[6 \, C_{\beta 0} - 4 \, C_{\beta 1} - 3 \, C_{\beta 2} + 2 \, C_{\beta 3} \right]$$

The differential equation

$$\frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = 0$$

is modeled with a general spatial marching scheme (after Ref. C-2)

$$\begin{aligned} &\frac{1}{\Delta x} \left[(1 + \epsilon) \, E_{i+1,j} - (1 + 2\epsilon) \, E_{ij} + \epsilon \, E_{i-1,j} \right] \\ &+ \frac{\theta}{\Delta y} \left[a \, F_{i+1,j+1} + (1 - 2a) \, F_{i+1,j} + (a-1) \, F_{i+1,j-1} \right] \\ &+ \frac{(1 - \theta + \phi)}{\Delta y} \left[a \, F_{i,j+1} + (1 - 2a) \, F_{i,j} + (a - 1) \, F_{i,j-1} \right] \end{aligned} \qquad \text{(C.5)}$$

$$- \frac{\phi}{\Delta y} \left[a F_{i-1,j+1} + (1 - 2a) F_{i-1,j} + (a-1) F_{i-1,j-1} \right]$$

(C.5)
(Concl'd)

$$= 0$$

Examination of Eq. (C.5) reveals the significance of the difference parameters:

a	cross-plane (η_2) difference parameter
	$= 1$ for forward cross-plane differences
	$= 0$ for backward cross-plane differences
	$= \frac{1}{2}$ for centered cross-plane differences
ϵ	marching (η_1) difference parameter
	$= 0$ for forward differences
	$= -1$ for backward differences
	$= -\frac{1}{2}$ for centered differences
θ	weighting parameter for plane (i+1)
	> 0 cross-plane differences are taken in the i^{th} and (i+1) st plane
	$= 0$ cross plane differences are not taken in the (i+1) st plane
ϕ	weighting parameter for plane (i-1)
	> 0 cross-plane differences are taken in the i^{th} and (i-1) st plane
	$= 0$ cross-plane differences are not taken in the (i-1) st plane

Note that for the present explicit differences in the elliptic GIM code $\theta = \phi = 0$.

Assuming a form of the weight functions similar to that presently in the code, i.e.,

$$W_\beta = \frac{d_{\beta 0}}{\Delta_J} (d_{\beta 1} - \eta_1)(d_{\beta 2} - \eta_2) \qquad \text{(C.6a)}$$

results in

$$d_{\beta 0} = 36\,(f_{\beta 1} - f_{\beta 2})\,(g_{\beta 1} - g_{\beta 2})/(f_{\beta 1} + f_{\beta 2})$$

$$d_{\beta 1} = \frac{1}{3}\left[(2\,g_{\beta 1} - g_{\beta 2})/(g_{\beta 1} - g_{\beta 2})\right] \qquad \text{(C.6b)}$$

$$d_{\beta 2} = \frac{1}{3}\left[(2\,f_{\beta 1} - f_{\beta 2})/(f_{\beta 1} - f_{\beta 2})\right]$$

and

$$f_{\beta 1} + f_{\beta 2} = g_{\beta 1} + g_{\beta 2}$$

Assembling the elements as in Ref. C-1 in terms f's and g's, equating these to the difference coefficients in Eq. (C.5) and substituting into Eqs. (C.6) yields weight functions of the following form[*]

$$W_1 = \frac{\alpha_1}{\Delta_J}(\beta_1 - \eta_1)(2/3 - \eta_2)$$

$$W_2 = \frac{\alpha_2}{\Delta_J}(\beta_2 - \eta_1)(2/3 - \eta_2)$$

$$W_3 = \frac{\alpha_3}{\Delta_J}(\beta_2 - \eta_1)(1/3 - \eta_2) \qquad \text{(C.7a)}$$

$$W_4 = \frac{\alpha_4}{\Delta_J}(\beta_1 - \eta_1)(1/3 - \eta_2)$$

where

$$\alpha_1 = 36\,a\,(1 + \epsilon - 2\theta) \qquad \text{(C.7b)}$$

$$\alpha_2 = 36\,a\,(\epsilon - 2\phi)$$

[*]Note that α's and β's are substituted here for the $d_{\beta i}$ in Eqs. (C.6) to show the similarity of these to the current GIM weight functions.

$$\alpha_3 = 36\,(a-1)\,(\epsilon - 2\phi)$$

$$\alpha_4 = 36\,(a-1)\,(1 + \epsilon - 2\theta) \qquad \text{(C.7b) (Concl'd)}$$

and

$$\beta_1 = \frac{1}{3}\left[\frac{2\,(1+\epsilon) - 3\theta}{1 + \epsilon - 2\theta}\right]$$

$$\beta_2 = \frac{1}{3}\left[\frac{\epsilon - 3\phi}{\epsilon - 2\phi}\right] \qquad \text{(C.7c)}$$

These apply except in the case where

$$\epsilon = 2\theta - 1; \qquad \theta \neq 0$$

(e.g., Crank-Nicholson or $\epsilon = 0$, $\theta = 1/2$). In these cases the weight functions do not maintain the same symmetry as the present weight functions. Values of the α's and β's are available from the author.

C.3 THREE-DIMENSIONAL TRILINEAR WEIGHT FUNCTIONS

The procedure for the three-dimensional case is much the same as the two-dimensional problem. We assume the same trilinear shape functions now in the GIM code:

$$S_\alpha = \left\{\begin{array}{c} (1-\eta_1)\,(1-\eta_2)\,(1-\eta_3) \\ \eta_1\,(1-\eta_2)\,(1-\eta_3) \\ \eta_1\,\eta_2\,(1-\eta_3) \\ (1-\eta_1)\,\eta_2\,(1-\eta_3) \\ (1-\eta_1)\,(1-\eta_2)\,\eta_3 \\ \eta_1\,(1-\eta_2)\,\eta_3 \\ \eta_1\,\eta_2\,\eta_3 \\ (1-\eta_1)\,\eta_2\,\eta_3 \end{array}\right. \qquad \text{(C.8)}$$

The weight functions are assumed to have the form:

$$\Delta_J \cdot W_\beta = C_{\beta 0} - C_{\beta 1}\eta_1 - C_{\beta 2}\eta_2 - C_{\beta 3}\eta_3 + C_{\beta 4}\eta_1\eta_2 + C_{\beta 5}\eta_1\eta_3 + C_{\beta 6}\eta_2\eta_3 - C_{\beta 7}\eta_1\eta_2\eta_3 \qquad \text{(C.9a)}$$

or

$$W_\beta = \frac{d_{\beta 0}}{\Delta J}(d_{\beta 1} - \eta_1)(d_{\beta 2} - \eta_2)(d_{\beta 3} - \eta_3) \qquad \text{(C.9b)}$$

The relations between the C's and the d's is obvious. The numbering system for the rectilinear element and the 27 node box are given in Fig. C-2.

The full expressions for the element difference operators are given in Ref. C-3. For the rectilinear box, where

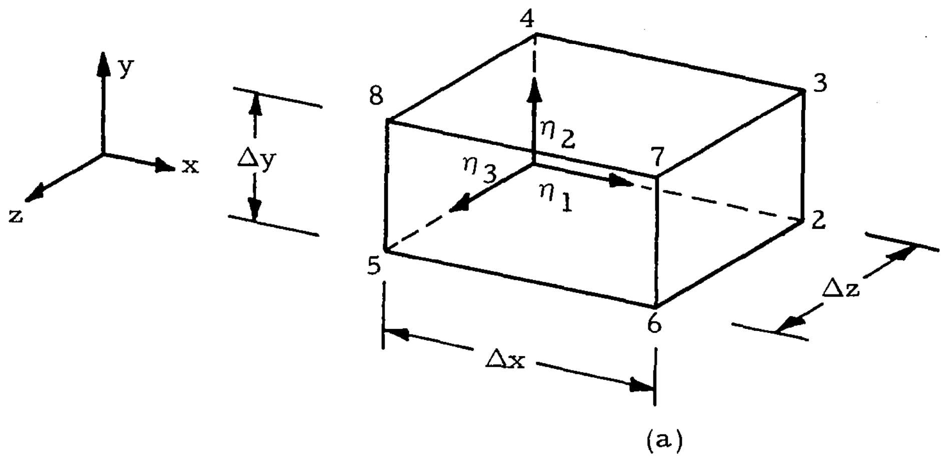
$$\Delta_J = \Delta x\, \Delta y\, \Delta z,\ \frac{\partial x}{\partial \eta_1} = \Delta x,\ \frac{\partial y}{\partial \eta_2} = \Delta y,\ \frac{\partial z}{\partial \eta_3} = \Delta z,\ \frac{\partial x_i}{\partial \eta_j} = 0 \quad i \neq j,$$

These operators become

$$\begin{aligned} D^{(1)}_{\beta\alpha} &= \Delta y\, \Delta z \int_0^1 d\eta_1 \int_0^1 d\eta_2 \int_0^1 d\eta_3\, W_\beta \frac{\partial S_\alpha}{\partial \eta_1} \\ D^{(2)}_{\beta\alpha} &= \Delta x\, \Delta z \int_0^1 d\eta_1 \int_0^1 d\eta_2 \int_0^1 d\eta_3\, W_\beta \frac{\partial S_\alpha}{\partial \eta_2} \qquad \text{(C.10)} \\ D^{(3)}_{\beta\alpha} &= \Delta x\, \Delta y \int_0^1 d\eta_1 \int_0^1 d\eta_2 \int_0^1 d\eta_3\, W_\beta \frac{\partial S_\alpha}{\partial \eta_3} \end{aligned}$$

Substituting Eqs. (C.8) and (C.9a) into Eq. (C-10) results in four unique difference operators for each direction

$$f_{\beta 1} \equiv \Delta x \cdot D^{(1)}_{\beta 2} = -\Delta x \cdot D^{(1)}_{\beta 1};\ g_{\beta 1} \equiv \Delta y \cdot D^{(2)}_{\beta 4} = -\Delta y \cdot D^{(2)}_{\beta 1};\ h_{\beta 1} \equiv \Delta z \cdot D^{(3)}_{\beta 5} = -\Delta z \cdot D^{(3)}_{\beta 1}$$



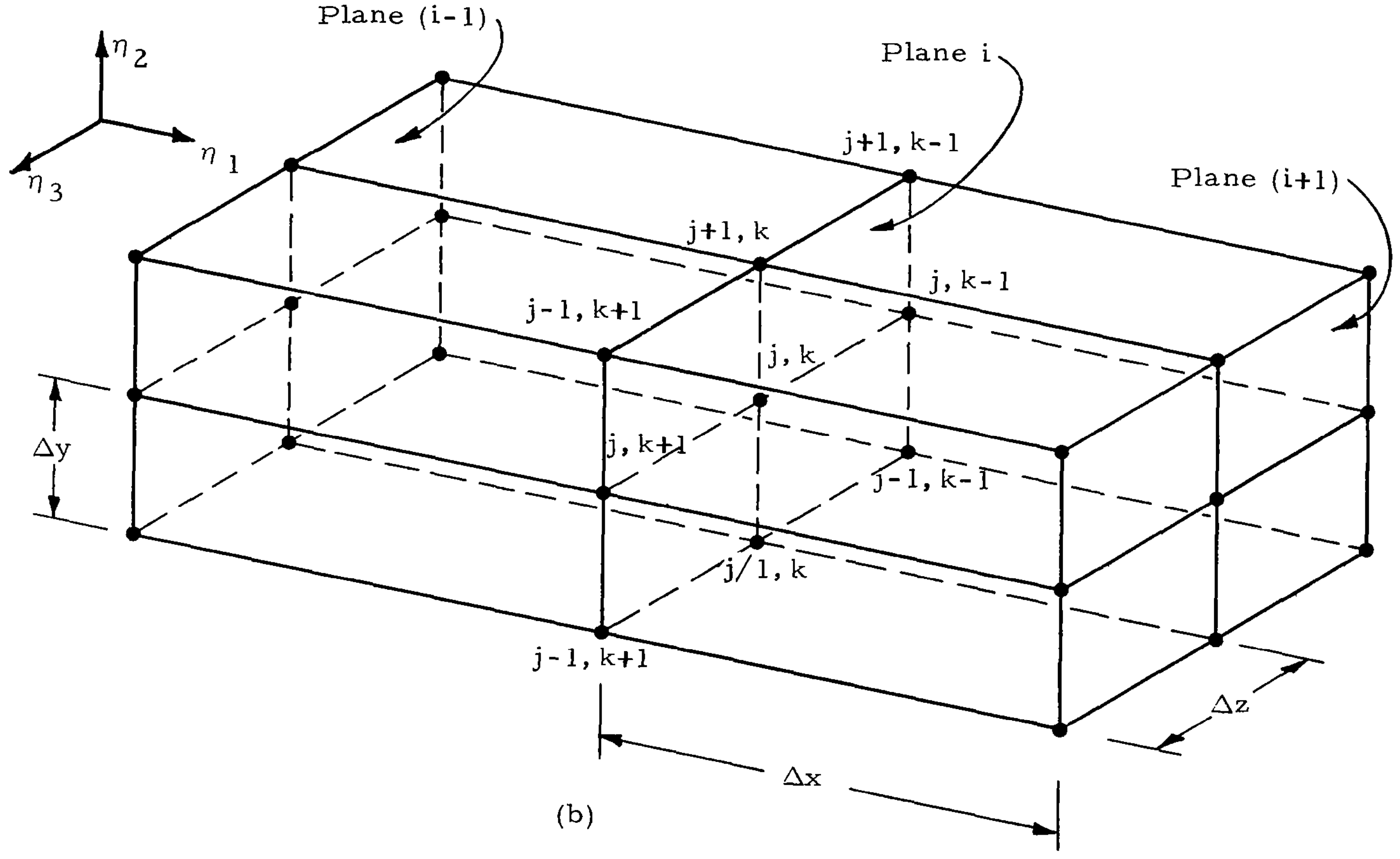


Fig. C-2 - Three-Dimensional Nodal Numbering System
(a) Element System; (b) 27-Node Rectilinear Box

It can be seen from the above that

$$f_{\beta 1} + f_{\beta 2} + f_{\beta 3} + f_{\beta 4} = g_{\beta 1} + g_{\beta 2} + g_{\beta 3} + g_{\beta 4} = h_{\beta 1} + h_{\beta 2} + h_{\beta 3} + h_{\beta 4} \qquad \text{(C.12a)}$$

and

$$\begin{aligned} f_{\beta 1} + f_{\beta 2} &= g_{\beta 1} + g_{\beta 2}; \quad f_{\beta 3} + f_{\beta 4} = g_{\beta 3} + g_{\beta 4} \\ f_{\beta 1} + f_{\beta 3} &= h_{\beta 1} + h_{\beta 2}; \quad f_{\beta 2} + f_{\beta 4} = h_{\beta 3} + h_{\beta 4} \qquad \text{(C.12b)} \\ g_{\beta 1} + g_{\beta 3} &= h_{\beta 1} + h_{\beta 4}; \quad g_{\beta 2} + g_{\beta 4} = h_{\beta 2} + h_{\beta 3} \end{aligned}$$

Now, Eqs. (C.9b), (C.11) and (C.12) can be combined to yield

$$\frac{f_{\beta 1}}{f_{\beta 2}} = \frac{f_{\beta 3}}{f_{\beta 4}}; \quad \frac{g_{\beta 1}}{g_{\beta 2}} = \frac{g_{\beta 3}}{g_{\beta 4}}; \quad \frac{h_{\beta 1}}{h_{\beta 2}} = \frac{h_{\beta 4}}{h_{\beta 3}} \qquad \text{(C.13a)}$$

$$d_{\beta 0} = 216 \left[\frac{(f_{\beta 1} + f_{\beta 2}) - (f_{\beta 3} + f_{\beta 4})}{f_{\beta 1} + f_{\beta 2} + f_{\beta 3} + f_{\beta 4}}\right]\left[\frac{(h_{\beta 1} + h_{\beta 2}) - (h_{\beta 3} + h_{\beta 4})}{h_{\beta 1} + h_{\beta 2} + h_{\beta 3} + h_{\beta 4}}\right]\left[(g_{\beta 1} + g_{\beta 3}) - (g_{\beta 2} + g_{\beta 4})\right]$$

$$\begin{aligned} d_{\beta 1} &= \frac{1}{3}\left[2(g_{\beta 1} + g_{\beta 3}) - (g_{\beta 2} + g_{\beta 4})\right] \Big/ \left[(g_{\beta 1} + g_{\beta 3}) - (g_{\beta 2} + g_{\beta 4})\right] \\ d_{\beta 2} &= \frac{1}{3}\left[2(h_{\beta 1} + h_{\beta 2}) - (h_{\beta 3} + h_{\beta 4})\right] \Big/ \left[(h_{\beta 1} + h_{\beta 2}) - (h_{\beta 3} + h_{\beta 4})\right] \qquad \text{(C.13b)} \\ d_{\beta 3} &= \frac{1}{3}\left[2(f_{\beta 1} + f_{\beta 2}) - (f_{\beta 3} + f_{\beta 4})\right] \Big/ \left[(f_{\beta 1} + f_{\beta 2}) - (f_{\beta 3} + f_{\beta 4})\right] \end{aligned}$$

The differential equation

$$\frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0$$

is modeled by

$$\frac{1}{\Delta x}\left[(1+\epsilon)\,E^{i+1}_{j,k} - (1+2\epsilon)\,E^{i}_{j,k} + \epsilon\,E^{i-1}_{j,k}\right]$$

$$+ \frac{\theta}{\Delta y}\left[a\,F^{i+1}_{j+1,k} + (1-2a)\,F^{i+1}_{i,k} + (a-1)\,F^{i+1}_{j-1,k}\right]$$

$$+ \frac{(1-\theta+\phi)}{\Delta y}\left[a\,F^{i}_{j+1,k} + (1-2a)\,F^{i}_{j,k} + (a-1)\,F^{i}_{j-1,k}\right]$$

$$- \frac{\phi}{\Delta y}\left[a\,F^{i-1}_{j+1,k} + (1-2a)\,F^{i-1}_{j,k} + (a-1)\,F^{i-1}_{j-1,k}\right] \qquad \text{(C.14)}$$

$$+ \frac{\theta}{\Delta z}\left[b\,F^{i+1}_{j,k+1} + (1-2b)\,G^{i+1}_{j,k} + (b-1)\,G^{i+1}_{j,k-1}\right]$$

$$+ \frac{(1-\theta+\phi)}{\Delta z}\left[b\,G^{i}_{j,k+1} + (1-2b)\,G^{i}_{j,k} + (b-1)\,G^{i}_{j,k-1}\right]$$

$$- \frac{\phi}{\Delta z}\left[b\,G^{i-1}_{j,k+1} + (1-2b)\,G^{i-1}_{j,k} + (b-1)\,G^{i-1}_{j,k-1}\right] = 0$$

where the difference parameters have the same significance as before and b is equivalent to a for differences in η_3.

Assembling the elements and equating coefficients does not lead to expressions for the weight functions in as straightforward a manner as in the two-dimensional problem. In order to resolve several ambiguities, the following considerations along with Eqs. (C.13a) were used:

1. The weight functions should reduce to the form presently in the code for $\theta = \phi = 0$.
2. The first four weight functions should readily reduce to the two-dimensional case. That is, the internal symmetries of the two-dimensional element difference operators should carry over to three dimensions.
3. The weight functions derived here should be applicable to the elliptic solver with $\theta = \phi = 0$. Thus, the boundary terms must be differenced consistantly.

From this, the following weight functions result:

$$W_1 = \frac{\alpha_1}{\Delta_J}(\beta_1 - \eta_1)\left(\frac{2}{3} - \eta_2\right)\left(\frac{2}{3} - \eta_3\right)$$

$$W_2 = \frac{\alpha_2}{\Delta_J}(\beta_2 - \eta_1)\left(\frac{2}{3} - \eta_2\right)\left(\frac{2}{3} - \eta_3\right)$$

$$W_3 = \frac{\alpha_3}{\Delta_J}(\beta_2 - \eta_1)\left(\frac{1}{3} - \eta_2\right)\left(\frac{2}{3} - \eta_3\right)$$

$$W_4 = \frac{\alpha_4}{\Delta_J}(\beta_1 - \eta_1)\left(\frac{1}{3} - \eta_2\right)\left(\frac{2}{3} - \eta_3\right)$$

$$W_5 = \frac{\alpha_5}{\Delta_J}(\beta_1 - \eta_1)\left(\frac{2}{3} - \eta_2\right)\left(\frac{1}{3} - \eta_3\right)$$

$$W_6 = \frac{\alpha_6}{\Delta_J}(\beta_2 - \eta_1)\left(\frac{2}{3} - \eta_2\right)\left(\frac{1}{3} - \eta_3\right)$$

$$W_7 = \frac{\alpha_7}{\Delta_J}(\beta_2 - \eta_1)\left(\frac{1}{3} - \eta_2\right)\left(\frac{1}{3} - \eta_3\right)$$

$$W_8 = \frac{\alpha_8}{\Delta_J}(\beta_1 - \eta_1)\left(\frac{1}{3} - \eta_2\right)\left(\frac{1}{3} - \eta_3\right)$$

where

$$\alpha_1 = (a - b + b^2)(1 + \epsilon - 2\theta); \quad \alpha_5 = b(b - 1)(1 + \epsilon - 2\theta)$$

$$\alpha_2 = (a - b + b^2)(\epsilon - 2\phi) \quad ; \quad \alpha_6 = b(b - 1)(\epsilon - 2\phi)$$

$$\alpha_3 = (\alpha - 2b + b^2)(\epsilon - 2\phi) \quad ; \quad \alpha_7 = (1 - b)^2(\epsilon - 2\phi)$$

$$\alpha_4 = (\alpha - 2b + b^2)(1 + \epsilon - 2\theta); \quad \alpha_8 = (1 - b)^2(1 + \epsilon - 2\theta)$$

and β_1 and β_2 are defined in Eq. (C.7c).

C.4 REFERENCES

C-1. Spradley, L. W., P. G. Anderson, and M. L. Pearson, "Computation of Three-Dimensional Nozzle-Exhaust Flow Fields with the GIM Code," NASA CR-3042 (1978).

C-2. Beam, R. M., and R. F. Warming, "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations; II, the Numerical ODE Connection," AIAA Paper No. 79-1446 (1979).

C-3. Spradley, L. W., and M. L. Pearson, "GIM Code User's Manual for the STAR-100 Computer," NASA CR-3157 (1979).

Appendix D

VECTORIZED BLOCK TRIDIAGONAL EQUATION SOLVER FOR THE GIM/STAR CODE

by

S. J. Robertson

Appendix D

An attempt was made to develop a vectorized algorithm for solving large systems of linear equations of the form:

$$L_{i-1}U_{i-1} + M_iU_i + N_{i+1}U_{i+1} = D_i \tag{D.1}$$

where the L, M and N elements are 3 x 3, 4 x 4 or 5 x 5 matrix blocks, and the U and D elements are three-, four- or five-component column vectors. The subscript i in Eq. (D.1) corresponds to nodal points in a computational grid, and the three, four or five dimensionality of the matrix and vector elements depend on whether the system of equations are for a one-, two- or three-dimensional flow field problem (see Section 3). The system of linear equations represented by Eq. (D.1) forms a block tridiagonal system.

A solution algorithm was sought that would make use of the parallel processing capability of the STAR-100 vector computer. The Gauss-Seidel relaxation technique, based on an iterated solution of

$$U_i^{k+1} = (1 - w)U_i^k - wM_i^{-1}(L_{i-1}U_{i-1}^k + N_{i+1}U_{i+1}^k - D_i) \tag{D.2}$$

where w is an over-relaxation factor, is the only technique which we could find that permits a straightforward use of vectorized computer programming. The inverse matrix M_i^{-1} in Eq. (D.2) is evaluated for all M_i prior to entering the iteration loop. Since each matrix block is dimensioned only up to 5 x 5, the inverse can be evaluated by direct algebraic manipulation or by a Gauss elimination technique. For the time being, we have coded only the algebraic inversion, since vector programming can be used in this method.

Separate subroutines were programmed for 3 x 3, 4 x 4 and 5 x 5 block tridiagonal Gauss-Seidel equation solvers. These are listed in Tables D-1, D-2 and D-3 as subroutines EQSOL3, EQSOL4 and EQSOL5, respectively. The argument list in these subroutines is (U, L, M, N, D, NODES, W, EPS, MAXI). The vector U is the solution vector which enters the subroutine as an initial or trial solution and returns as the updated or final solution. The matrices L, M and N and the vector D enter the subroutine as constants. The scalar NODES is the number of nodal points, W is the over-relaxation parameter, EPS is the error tolerance in the convergence test and MAXI is the maximum allowable iterations. The U and D vectors are doubly dimensioned, and the matrices L, M and N are triply dimensioned. The first subscript of both vectors and matrices corresponds to the nodal point index. The second subscript of U and D corresponds to the vector components, and the second and third subscripts of L, M and N corresponds to the matrix elements.

As of this writing, these subroutines have not been evaluated, except for some very simple test cases. They have not been applied to realistic fluid dynamics problems where their usefulness can be determined.

Table D-1
LIST OF SUBROUTINE EQSOL3

```
      SUBROUTINE EQSOL3(U,L,M,N,D,NODES,W,EPS,MAXI)
      REAL L,M,MI,N
      DIMENSION U(9,3),L(9,3,3),M(9,3,3),MI(9,3,3),
     $ N(9,3,3),D(9,3),DET(9),UP(9,3),DIF(9,3)
      COMMON/XMI/MI
      MI(1,1,1$NODES)=M(1,2,2$NODES)* M(1,3,3$NODES)
     $                                 M(1,3,2$NODES)* M(1,2,3$NODES)
      MI(1,2,1$NODES)=-M(1,2,1$NODES)* M(1,3,3$NODES)
     $                                +M(1,3,1$NODES)* M(1,2,3$NODES)
      MI(1,3,1$NODES)=M(1,2,1$NODES)* M(1,3,2$NODES)
     $                                 M(1,3,1$NODES)* M(1,2,2$NODES)
      MI(1,1,2$NODES)=-M(1,1,2$NODES)* M(1,3,3$NODES)
     $                                +M(1,3,2$NODES)* M(1,1,3$NODES)
      MI(1,2,2$NODES)=M(1,1,1$NODES)* M(1,3,3$NODES)
     $                                 M(1,3,1$NODES)* M(1,1,3$NODES)
      MI(1,3,2$NODES)=-M(1,1,1$NODES)* M(1,3,2$NODES)
     $                                +M(1,3,1$NODES)* M(1,1,2$NODES)
      MI(1,1,3$NODES)=M(1,1,2$NODES)* M(1,2,3$NODES)
     $                                -M(1,2,2$NODES)* M(1,1,3$NODES)
      MI(1,2,3$NODES)=-M(1,1,1$NODES)* M(1,2,3$NODES)
     $                                +M(1,2,1$NODES)* M(1,1,3$NODES)
      MI(1,3,3$NODES)=M(1,1,1$NODES)* M(1,2,2$NODES)
     $                                 M(1,2,1$NODES)* M(1,1,2$NODES)
      DET(1$NODES)=M(1,1,1$NODES)*MI(1,1,1$NODES)
     $             +M(1,1,2$NODES)*MI(1,2,1$NODES)
     $             +M(1,1,3$NODES)*MI(1,3,1$NODES)
      DO 50 I=1,3
      DO 50 J=1,3
      MI(1,I,J$NODES)=MI(1,I,J$NODES)/DET(1$NODES)
   50 CONTINUE
      NM1=NODES-1
      NM2=NODES-2
      ITER=0
   10 CONTINUE
      ITER=ITER+1
      DO 100 I=1,3
      UP(1,I)=(1.-W)*U(1,I)
      UP(1,I)=UP(1,I)-W*MI(1,I,1)*(N(1,1,1)*U(2,1)+N(1,1,2)*U(2,2)
     $ +N(1,1,3)*U(2,3))
```

(Continued)

Table D-1 (Concluded)

```
      UP(1,I)=UP(1,I)-W*MI(1,I,2)*(N(1,2,1)*U(2,1)+N(1,2,2)*U(2,2)
     $ +N(1,2,3)*U(2,3))
      UP(1,I)=UP(1,I)-W*MI(1,I,3)*(N(1,3,1)*U(2,1)+N(1,3,2)*U(2,2)
     $ +N(1,3,3)*U(2,3))
      UP(1,I)=UP(1,I)+W*(MI(1,I,1)*D(1,1)+MI(1,I,2)*D(1,2)+
     $ MI(1,I,3)*D(1,3))
      UP(2,I$NM2)=(1.-W)*U(2,I$NM2)
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,1$NM2)*(L(2,1,1$NM2)*U(1,1$NM2)
     $ +L(2,1,2$NM2)*U(1,2$NM2)+L(2,1,3$NM2)*U(1,3$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,1$NM2)*(N(2,1,1$NM2)*U(3,1$NM2)
     $ +N(2,1,2$NM2)*U(3,2$NM2)+N(2,1,3$NM2)*U(3,3$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,2$NM2)*(L(2,2,1$NM2)*U(1,1$NM2)
     $ +L(2,2,2$NM2)*U(1,2$NM2)+L(2,2,3$NM2)*U(1,3$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,2$NM2)*(N(2,2,1$NM2)*U(3,1$NM2)
     $ +N(2,2,2$NM2)*U(3,2$NM2)+N(2,2,3$NM2)*U(3,3$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,3$NM2)*(L(2,3,1$NM2)*U(1,1$NM2)
     $ +L(2,3,2$NM2)*U(1,2$NM2)+L(2,3,3$NM2)*U(1,3$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,3$NM2)*(N(2,3,1$NM2)*U(3,1$NM2)
     $ +N(2,3,2$NM2)*U(3,2$NM2)+N(2,3,3$NM2)*U(3,3$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)+W*(MI(2,I,1$NM2)*D(2,1$NM2)+MI(2,I,2$NM2)
     $ *D(2,2$NM2)+MI(2,I,3$NM2)*D(2,3$NM2)                    )
      UP(NODES,I)=(1.-W)*U(NODES,I)
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,1)*(L(NODES,1,1)*U(NM1,1)
     $ +L(NODES,1,2)*U(NM1,2)+L(NODES,1,3)*U(NM1,3)  )
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,2)*(L(NODES,2,1)*U(NM1,1)
     $ +L(NODES,2,2)*U(NM1,2)+L(NODES,2,3)*U(NM1,3))
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,3)*(L(NODES,3,1)*U(NM1,1)
     $ +L(NODES,3,2)*U(NM1,2)+L(NODES,3,3)*U(NM1,3))
      UP(NODES,I)=UP(NODES,I)+W*(MI(NODES,I,1)*D(NODES,1)+MI(NODES,I,2)
     $ *D(NODES,2)+MI(NODES,I,3)*D(NODES,3))
100   CONTINUE
      DO 350 I=1,3
      DIF(1,I$NODES)= UP(1,I$NODES)-U(1,I$NODES)
350   CONTINUE
      RMS=0.
      DEL=0.
      DO 360 I=1,3
      DO 360 J=1,NODES
      DEL=DEL+DIF(J,I)*DIF(J,I)
      RMS=RMS+UP(J,I)*UP(J,I)
360   CONTINUE
      DEL=SQRT(DEL)
      RMS=SQRT(RMS)
      TEST=DEL/RMS
      DO 400 I=1,3
      U(1,I$NODES)=UP(1,I$NODES)
400   CONTINUE
      IF(TEST.LE.EPS.OR.ITER.GE.MAXI)RETURN
      GO TO 10
      END
```

Table D-2
LISTING OF SUBROUTINE EQSOL4

```
      SUBROUTINE EQSOL4(U,L,M,N,D,NODES,W,EPS,MAXI)
      REAL L,M,MI,N
      DIMENSION U(9,4),L(9,4,4),M(9,4,4),MI(9,4,4),
     $ N(9,4,4),D(9,4),DET(9),UP(9,4),DIF(9,4)
      COMMON/XMI/MI
      MI(1,1,1$NODES)=M(1,2,2$NODES)*(M(1,3,3$NODES)*M(1,4,4$NODES)
     $ - M(1,4,3$NODES)*M(1,3,4$NODES))-M(1,3,2$NODES)*(M(1,2,3$NODES)
     $ *M(1,4,4$NODES)-M(1,4,3$NODES)*M(1,2,4$NODES))+M(1,4,2$NODES)
     $ *(M(1,2,3$NODES)*M(1,3,4$NODES)-M(1,3,3$NODES)*M(1,2,4$NODES))
      MI(1,2,1$NODES)=-M(1,2,1$NODES)*(M(1,3,3$NODES)*M(1,4,4$NODES)
     $ -M(1,4,3$NODES)*M(1,3,4$NODES))+M(1,3,1$NODES)*(M(1,2,3$NODES)
     $ *M(1,4,4$NODES)-M(1,4,3$NODES)*M(1,2,4$NODES))-M(1,4,1$NODES)
     $ *(M(1,2,3$NODES)*M(1,3,4$NODES)-M(1,3,3$NODES)*M(1,2,4$NODES))
      MI(1,3,1$NODES)=M(1,2,1$NODES)*(M(1,3,2$NODES)*M(1,4,4$NODES)
     $ -M(1,4,2$NODES)*M(1,3,4$NODES))-M(1,3,1$NODES)*(M(1,2,2$NODES)
     $ *M(1,4,4$NODES)-M(1,4,2$NODES)*M(1,2,4$NODES))+M(1,4,1$NODES)
     $ *(M(1,2,2$NODES)*M(1,3,4$NODES)-M(1,3,2$NODES)*M(1,2,4$NODES))
      MI(1,4,1$NODES)=-M(1,2,1$NODES)*(M(1,3,2$NODES)*M(1,4,3$NODES)
     $-M(1,4,2$NODES)*M(1,3,3$NODES))+M(1,3,1$NODES)*(M(1,2,2$NODES)
     $ *M(1,4,3$NODES)-M(1,4,2$NODES)*M(1,2,3$NODES))-M(1,4,1$NODES)
     $ *(M(1,2,2$NODES)*M(1,3,3$NODES)-M(1,3,2$NODES)*M(1,2,3$NODES))
      MI(1,1,2$NODES)=-M(1,1,2$NODES)*(M(1,3,3$NODES)*M(1,4,4$NODES)
     $ -M(1,4,3$NODES)*M(1,3,4$NODES))+M(1,3,2$NODES)*(M(1,1,3$NODES)
     $ *M(1,4,4$NODES)-M(1,4,3$NODES)*M(1,1,4$NODES))-M(1,4,2$NODES)
     $ *(M(1,1,3$NODES)*M(1,3,4$NODES)-M(1,3,3$NODES)*M(1,1,4$NODES))
      MI(1,2,2$NODES)=M(1,1,1$NODES)*(M(1,3,3$NODES)*M(1,4,4$NODES)
     $ -M(1,4,3$NODES)*M(1,3,4$NODES))-M(1,3,1$NODES)*(M(1,1,3$NODES)
     $ *M(1,4,4$NODES)-M(1,4,3$NODES)*M(1,1,4$NODES))+M(1,4,1$NODES)
     $ *(M(1,1,3$NODES)*M(1,3,4$NODES)-M(1,3,3$NODES)*M(1,1,4$NODES))
      MI(1,3,2$NODES)=-M(1,1,1$NODES)*(M(1,3,2$NODES)*M(1,4,4$NODES)
     $ -M(1,4,2$NODES)*M(1,3,4$NODES))+M(1,3,1$NODES)*(M(1,1,2$NODES)
     $ *M(1,4,4$NODES)-M(1,4,2$NODES)*M(1,1,4$NODES))-M(1,4,1$NODES)
     $ *(M(1,1,2$NODES)*M(1,3,4$NODES)-M(1,3,2$NODES)*M(1,1,4$NODES))
      MI(1,4,2$NODES)=M(1,1,1$NODES)*(M(1,3,2$NODES)*M(1,4,3$NODES)
     $ -M(1,4,2$NODES)*M(1,3,3$NODES))-M(1,3,1$NODES)*(M(1,1,2$NODES)
     $ *M(1,4,3$NODES)-M(1,4,2$NODES)*M(1,1,3$NODES))+M(1,4,1$NODES)
     $ *(M(1,1,2$NODES)*M(1,3,3$NODES)-M(1,3,2$NODES)*M(1,1,3$NODES))
      MI(1,1,3$NODES)=M(1,1,2$NODES)*(M(1,2,3$NODES)*M(1,4,4$NODES)
     $ -M(1,4,3$NODES)*M(1,2,4$NODES))-M(1,2,2$NODES)*(M(1,1,3$NODES)
     $ *M(1,4,4$NODES)-M(1,4,3$NODES)*M(1,1,4$NODES))+M(1,4,2$NODES)
     $ *(M(1,1,3$NODES)*M(1,2,4$NODES)-M(1,2,3$NODES)*M(1,1,4$NODES))
      MI(1,2,3$NODES)=-M(1,1,1$NODES)*(M(1,2,3$NODES)*M(1,4,4$NODES)
```

(Continued)

Table D-2 (Continued)

```
$  -M(1,4,3$NODES)*M(1,2,4$NODES))+M(1,2,1$NODES)*(M(1,1,3$NODES)
$  *M(1,4,4$NODES)-M(1,4,3$NODES)*M(1,1,4$NODES))-M(1,4,1$NODES)
$  *(M(1,1,3$NODES)*M(1,2,4$NODES)-M(1,2,3$NODES)*M(1,1,4$NODES))
   MI(1,3,3$NODES)=M(1,1,1$NODES)*(M(1,2,2$NODES)*M(1,4,4$NODES)
$  -M(1,4,2$NODES)*M(1,2,4$NODES))-M(1,2,1$NODES)*(M(1,1,2$NODES)
$  *M(1,4,4$NODES)-M(1,4,2$NODES)*M(1,1,4$NODES))+M(1,4,1$NODES)
$  *(M(1,1,2$NODES)*M(1,2,4$NODES)-M(1,2,2$NODES)*M(1,1,4$NODES))
   MI(1,4,3$NODES)=-M(1,1,1$NODES)*(M(1,2,2$NODES)*M(1,4,3$NODES)
$  -M(1,4,2$NODES)*M(1,2,3$NODES))+M(1,2,1$NODES)*(M(1,1,2$NODES)
$  *M(1,4,3$NODES)-M(1,4,2$NODES)*M(1,1,3$NODES))-M(1,4,1$NODES)
$  *(M(1,1,2$NODES)*M(1,2,3$NODES)-M(1,2,2$NODES)*M(1,1,3$NODES))
   MI(1,1,4$NODES)=-M(1,1,2$NODES)*(M(1,2,3$NODES)*M(1,3,4$NODES)
$  -M(1,3,3$NODES)*M(1,2,4$NODES))+M(1,2,2$NODES)*(M(1,1,3$NODES)
$  *M(1,3,4$NODES)-M(1,3,3$NODES)*M(1,1,4$NODES))-M(1,3,2$NODES)
$  *(M(1,1,3$NODES)*M(1,2,4$NODES)-M(1,2,3$NODES)*M(1,1,4$NODES))
   MI(1,2,4$NODES)=M(1,1,1$NODES)*(M(1,2,3$NODES)*M(1,3,4$NODES)
$  -M(1,3,3$NODES)*M(1,2,4$NODES))-M(1,2,1$NODES)*(M(1,1,3$NODES)
$  *M(1,3,4$NODES)-M(1,3,3$NODES)*M(1,1,4$NODES))+M(1,3,1$NODES)
$  *(M(1,1,3$NODES)*M(1,2,4$NODES)-M(1,2,3$NODES)*M(1,1,4$NODES))
   MI(1,3,4$NODES)=-M(1,1,1$NODES)*(M(1,2,2$NODES)*M(1,3,4$NODES)
$  -M(1,3,2$NODES)*M(1,2,4$NODES))+M(1,2,1$NODES)*(M(1,1,2$NODES)
$  *M(1,3,4$NODES)-M(1,3,2$NODES)*M(1,1,4$NODES))-M(1,3,1$NODES)
$  *(M(1,1,2$NODES)*M(1,2,4$NODES)-M(1,2,2$NODES)*M(1,1,4$NODES))
   MI(1,4,4$NODES)=M(1,1,1$NODES)*(M(1,2,2$NODES)*M(1,3,3$NODES)
$  -M(1,3,2$NODES)*M(1,2,3$NODES))-M(1,2,1$NODES)*(M(1,1,2$NODES)
$  *M(1,3,3$NODES)-M(1,3,2$NODES)*M(1,1,3$NODES))+M(1,3,1$NODES)
$  *(M(1,1,2$NODES)*M(1,2,3$NODES)-M(1,2,2$NODES)*M(1,1,3$NODES))
   DET(1$NODES)=M(1,1,1$NODES)*MI(1,1,1$NODES)
$                 +M(1,1,2$NODES)*MI(1,2,1$NODES)
$                 +M(1,1,3$NODES)*MI(1,3,1$NODES)
$                 +M(1,1,4$NODES)*MI(1,4,1$NODES)
   NM1=NODES-1
   NM2=NODES-2
   DO 50 I=1,4
   DO 50 J=1,4
   MI(1,I,J$NODES)=MI(1,I,J$NODES)/DET(1$NODES)
50 CONTINUE
   ITER=0
10 CONTINUE
   ITER=ITER+1
   DO 100 I=1,4
   UP(1,I)=(1.-W)*U(1,I)
   UP(1,I)=UP(1,I)-W*MI(1,I,1)*(N(2,1,1)*U(2,1)+N(2,1,2)*U(2,2)
$  +N(2,1,3)*U(2,3)+N(2,1,4)*U(2,4))
   UP(1,I)=UP(1,I)-W*MI(1,I,2)*(N(2,2,1)*U(2,1)+N(2,2,2)*U(2,2)
```

(Continued)

Table D-2 (Continued)

```
$ +N(2,2,3)*U(2,3)+N(2,2,4)*U(2,4))
  UP(1,I)=UP(1,I)-W*MI(1,I,3)*(N(2,3,1)*U(2,1)+N(2,3,2)*U(2,2)
$ +N(2,3,3)*U(2,3)+N(2,3,4)*U(2,4))
  UP(1,I)=UP(1,I)-W*MI(1,I,4)*(N(2,4,1)*U(2,1)+N(2,4,2)*U(2,2)
$ +N(2,4,3)*U(2,3)+N(2,4,4)*U(2,4))
  UP(1,I)=UP(1,I)+W*(MI(1,I,1)*D(1,1)+MI(1,I,2)*D(1,2)+
$ MI(1,I,3)*D(1,3)+MI(1,I,4)*D(1,4))
  UP(2,I$NM2)=(1.-W)*U(2,I$NM2)
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,1$NM2)*(L(1,1,1$NM2)*U(1,1$NM2)
$ +L(1,1,2$NM2)*U(1,2$NM2)+L(1,1,3$NM2)*U(1,3$NM2)+L(1,1,4$NM2)*
$ U(1,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,1$NM2)*(N(3,1,1$NM2)*U(3,1$NM2)
$ +N(3,1,2$NM2)*U(3,2$NM2)+N(3,1,3$NM2)*U(3,3$NM2)+N(3,1,4$NM2)*
$ U(3,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,2$NM2)*(L(1,2,1$NM2)*U(1,1$NM2)
$ +L(1,2,2$NM2)*U(1,2$NM2)+L(1,2,3$NM2)*U(1,3$NM2)+L(1,2,4$NM2)*
$ U(1,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,2$NM2)*(N(3,2,1$NM2)*U(3,1$NM2)
$ +N(3,2,2$NM2)*U(3,2$NM2)+N(3,2,3$NM2)*U(3,3$NM2)+N(3,2,4$NM2)*
$ U(3,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,3$NM2)*(L(1,3,1$NM2)*U(1,1$NM2)
$ +L(1,3,2$NM2)*U(1,2$NM2)+L(1,3,3$NM2)*U(1,3$NM2)+L(1,3,4$NM2)*
$ U(1,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,3$NM2)*(N(3,3,1$NM2)*U(3,1$NM2)
$ +N(3,3,2$NM2)*U(3,2$NM2)+N(3,3,3$NM2)*U(3,3$NM2)+N(3,3,4$NM2)*
$ U(3,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,4$NM2)*(L(1,4,1$NM2)*U(1,1$NM2)
$ +L(1,4,2$NM2)*U(1,2$NM2)+L(1,4,3$NM2)*U(1,3$NM2)+L(1,4,4$NM2)*
$ U(1,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,4$NM2)*(N(3,4,1$NM2)*U(3,1$NM2)
$ +N(3,4,2$NM2)*U(3,2$NM2)+N(3,4,3$NM2)*U(3,3$NM2)+N(3,4,4$NM2)*
$ U(3,4$NM2))
  UP(2,I$NM2)=UP(2,I$NM2)+W*(MI(2,I,1$NM2)*D(2,1$NM2)+MI(2,I,2$NM2)
$ *D(2,2$NM2)+MI(2,I,3$NM2)*D(2,3$NM2)+MI(2,I,4$NM2)*D(2,4$NM2))
  UP(NODES,I)=(1.-W)*U(NODES,I)
  UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,1)*(L(NM1    ,1,1)*U(NM1,1)
$ +L(NM1    ,1,2)*U(NM1,2)+L(NM1    ,1,3)*U(NM1,3)+L(NM1    ,1,4)
$ *U(NM1,4))
  UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,2)*(L(NM1    ,2,1)*U(NM1,1)
$ +L(NM1    ,2,2)*U(NM1,2)+L(NM1    ,2,3)*U(NM1,3)+L(NM1    ,2,4)
$ *U(NM1,4))
  UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,3)*(L(NM1    ,3,1)*U(NM1,1)
$ +L(NM1    ,3,2)*U(NM1,2)+L(NM1    ,3,3)*U(NM1,3)+L(NM1    ,3,4)
$ *U(NM1,4))
  UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,4)*(L(NM1    ,4,1)*U(NM1,1)
$ +L(NM1    ,4,2)*U(NM1,2)+L(NM1    ,4,3)*U(NM1,3)+L(NM1    ,4,4)
```

(Continued)

Table D-2 (Concluded)

```
     $ *U(NM1,4))
      UP(NODES,I)=UP(NODES,I)+W*(MI(NODES,I,1)*D(NODES,1)+MI(NODES,I,2
     $ *D(NODES,2)+MI(NODES,I,3)*D(NODES,3)+MI(NODES,I,4)*D(NODES,4))
100   CONTINUE
      DO 350 I=1,4
      DIF(1,I$NODES)= UP(1,I$NODES)-U(1,I$NODES)
350   CONTINUE
      RMS=0.
      DEL=0.
      DO 360 I=1,4
      DO 360 J=1,NODES
      DEL=DEL+DIF(J,I)*DIF(J,I)
      RMS=RMS+UP(J,I)*UP(J,I)
360   CONTINUE
      DEL=SQRT(DEL)
      RMS=SQRT(RMS)
      TEST=DEL/RMS
      WRITE(6,450)UP,J,DIF
450   FORMAT(9E10.3)
      WRITE(6,500) ITER,TEST,DEL,RMS
500   FORMAT(I5,3E10.3)
      DO 400 I=1,4
      U(1,I$NODES)=UP(1,I$NODES)
400   CONTINUE
      IF(TEST.LE.EPS.OR.ITER.GE.MAXI)RETURN
      GO TO 10
      END
```

Table D-3
LISTING OF SUBROUTINE EQSOL5

```
 SUBROUTINE EQSOL5(U,L,M,N,D,NODES,W,EPS,MAXI)
 DIMENSION U(9,5),L(9,5,5),M(9,5,5),MI(9,5,5),
$ N(9,5,5),D(9,5),DET(9),UP(9,5),DIF(9,5)
 REAL L,M,MI,N
 COMMON/XMI/MI
 MI(1,1,1$NODES)=(M(1,2,2$NODES)*M(1,3,3$NODES)-M(1,3,2$NODES)*
$ M(1,2,3$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,1,1$NODES)=MI(1,1,1$NODES)-(M(1,2,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,2,3$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)
$   -M(1,5,4$NODES)*M(1,3,5$NODES))
 MI(1,1,1$NODES)=MI(1,1,1$NODES)+(M(1,2,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,2,3$NODES'))*(M(1,3,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,3,5$NODES))
 MI(1,1,1$NODES)=MI(1,1,1$NODES)+(M(1,3,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,3,3$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
 MI(1,1,1$NODES)=MI(1,1,1$NODES)-(M(1,3,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,3,3$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
 MI(1,1,1$NODES)=MI(1,1,1$NODES)+(M(1,4,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,4,3$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
 MI(1,2,1$NODES)=(M(1,2,1$NODES)*M(1,3,3$NODES)-M(1,3,1$NODES)*
$ M(1,2,3$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,2,1$NODES)=MI(1,1,2$NODES)-(M(1,2,1$NODES)*M(1,4,3$NODES)
$ -M(1,4,1$NODES)*M(1,2,3$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,3,5$NODES))
 MI(1,2,1$NODES)=MI(1,1,2$NODES)+(M(1,2,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,2,3$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,3,5$NODES))
 MI(1,2,1$NODES)=MI(1,1,2$NODES)+(M(1,3,1$NODES)*M(1,4,3$NODES)
$ -M(1,4,1$NODES)*M(1,3,3$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
 MI(1,2,1$NODES)=MI(1,1,2$NODES)-(M(1,3,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,3,3$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$ NODES)*M(1,2,5$NODES))
 MI(1,2,1$NODES)=MI(1,1,2$NODES)+(M(1,4,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,4,3$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
 MI(1,3,1$NODES)=(M(1,2,1$NODES)*M(1,3,2$NODES)-M(1,3,1$NODES)*
```

(Continued)

Table D-3 (Continued)

```
$ M(1,2,2$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
  MI(1,3,1$NODES)=MI(1,1,3$NODES)-(M(1,2,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,3,5$NODES))
  MI(1,3,1$NODES)=MI(1,1,3$NODES)+(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,3,5$NODES))
  MI(1,3,1$NODES)=MI(1,1,3$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
  MI(1,3,1$NODES)=MI(1,1,3$NODES)-(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
  MI(1,3,1$NODES)=MI(1,1,3$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
  MI(1,4,1$NODES)=(M(1,2,1$NODES)*M(1,3,2$NODES)-M(1,3,1$NODES)*
$ M(1,2,2$NODES))*(M(1,4,3$NODES)*M(1,5,5$NODES)-M(1,5,3$NODES)*
$ M(1,4,5$NODES))
  MI(1,4,1$NODES)=MI(1,1,4$NODES)-(M(1,2,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,3,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,3,5$NODES))
  MI(1,4,1$NODES)=MI(1,1,4$NODES)+(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,3,3$NODES)*M(1,4,5$NODES)
$ -M(1,4,3$NODES)*M(1,3,5$NODES))
  MI(1,4,1$NODES)=MI(1,1,4$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,2,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,2,5$NODES))
  MI(1,4,1$NODES)=MI(1,1,4$NODES)-(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,2,3$NODES)*M(1,4,5$NODES)
$ -M(1,4,3$NODES)*M(1,2,5$NODES))
  MI(1,4,1$NODES)=MI(1,1,4$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,2,3$NODES)*M(1,3,5$NODES)
$ -M(1,3,3$NODES)*M(1,2,5$NODES))
  MI(1,5,1$NODES)=(M(1,2,1$NODES)*M(1,3,2$NODES)-M(1,3,1$NODES)*
$ M(1,2,2$NODES))*(M(1,4,3$NODES)*M(1,5,4$NODES)-M(1,5,3$NODES)*
$ M(1,4,4$NODES))
  MI(1,5,1$NODES)=MI(1,1,5$NODES)-(M(1,2,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,3,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,3,4$NODES))
  MI(1,5,1$NODES)=MI(1,1,5$NODES)+(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,3,3$NODES)*M(1,4,4$NODES)
$ -M(1,4,3$NODES)*M(1,3,4$NODES))
  MI(1,5,1$NODES)=MI(1,1,5$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,2,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,2,4$NODES))
```

(Continued)

Table D-3 (Continued)

```
 MI(1,5,1$NODES)=MI(1,1,5$NODES)-(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,2,3$NODES)*M(1,4,4$NODES)
$ -M(1,4,3$NODES)*M(1,2,4$NODES))
 MI(1,5,1$NODES)=MI(1,1,5$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,2,3$NODES)*M(1,3,4$NODES)
$ -M(1,3,3$NODES)*M(1,2,4$NODES))
 MI(1,1,2$NODES)=(M(1,1,2$NODES)*M(1,3,3$NODES)-M(1,3,2$NODES)*
$ M(1,1,3$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,1,2$NODES)=MI(1,2,1$NODES)-(M(1,1,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,3,5$NODES))
 MI(1,1,2$NODES)=MI(1,2,1$NODES)+(M(1,1,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,3,5$NODES))
 MI(1,1,2$NODES)=MI(1,2,1$NODES)+(M(1,3,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,1,2$NODES)=MI(1,2,1$NODES)-(M(1,3,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,1,2$NODES)=MI(1,2,1$NODES)+(M(1,4,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,4,3$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,1,5$NODES))
 MI(1,2,2$NODES)=(M(1,1,1$NODES)*M(1,3,3$NODES)-M(1,3,1$NODES)*
$ M(1,1,3$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,2,2$NODES)=MI(1,2,2$NODES)-(M(1,1,1$NODES)*M(1,4,3$NODES)
$   -M(1,4,1$NODES)*M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,3,5$NODES))
 MI(1,2,2$NODES)=MI(1,2,2$NODES)+(M(1,1,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,3,5$NODES))
 MI(1,2,2$NODES)=MI(1,2,2$NODES)+(M(1,3,1$NODES)*M(1,4,3$NODES)
$ -M(1,4,1$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,2,2$NODES)=MI(1,2,2$NODES)-(M(1,3,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,2,2$NODES)=MI(1,2,2$NODES)+(M(1,4,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,4,3$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,1,5$NODES))
 MI(1,3,2$NODES)=(M(1,1,1$NODES)*M(1,3,2$NODES)-M(1,3,1$NODES)*
$ M(1,1,2$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,3,2$NODES)=MI(1,2,3$NODES)-(M(1,1,1$NODES)*M(1,4,2$NODES)
```

(Continued)

Table D-3 (Continued)

```
$ -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,3,5$NODES))
 MI(1,3,2$NODES)=MI(1,2,3$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$ NODES)*M(1,1,2$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,3,5$NODES))
 MI(1,3,2$NODES)=MI(1,2,3$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,3,2$NODES)=MI(1,2,3$NODES)-(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,3,2$NODES)=MI(1,2,3$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,1,5$NODES))
 MI(1,4,2$NODES)=(M(1,1,1$NODES)*M(1,3,2$NODES)-M(1,3,1$NODES)*
$ M(1,1,2$NODES))*(M(1,4,3$NODES)*M(1,5,5$NODES)-M(1,5,3$NODES)*
$ M(1,4,5$NODES))
 MI(1,4,2$NODES)=MI(1,2,4$NODES)-(M(1,1,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,3,5$NODES))
 MI(1,4,2$NODES)=MI(1,2,4$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,4,5$NODES)
$ -M(1,4,3$NODES)*M(1,3,5$NODES))
 MI(1,4,2$NODES)=MI(1,2,4$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,1,5$NODES))
 MI(1,4,2$NODES)=MI(1,2,4$NODES)-(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,4,5$NODES)
$ -M(1,4,3$NODES)*M(1,1,5$NODES))
 MI(1,4,2$NODES)=MI(1,2,4$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,1,3$NODES)*M(1,3,5$NODES)
$ -M(1,3,3$NODES)*M(1,1,5$NODES))
 MI(1,5,2$NODES)=(M(1,1,1$NODES)*M(1,3,2$NODES)-M(1,3,1$NODES)*
$ M(1,1,2$NODES))*(M(1,4,3$NODES)*M(1,5,4$NODES)-M(1,5,3$NODES)*
$ M(1,4,4$NODES))
 MI(1,5,2$NODES)=MI(1,2,5$NODES)-(M(1,1,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,3,4$NODES))
 MI(1,5,2$NODES)=MI(1,2,5$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,4,4$NODES)
$ -M(1,4,3$NODES)*M(1,3,4$NODES))
 MI(1,5,2$NODES)=MI(1,2,5$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,1,4$NODES))
 MI(1,5,2$NODES)=MI(1,2,5$NODES)-(M(1,3,1$NODES)*M(1,5,2$NODES)
```

(Continued)

Table D-3 (Continued)

```
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,4,4$NODES)
$ -M(1,4,3$NODES)*M(1,1,4$NODES))
 MI(1,5,2$NODES)=MI(1,2,5$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,1,3$NODES)*M(1,3,4$NODES)
$ -M(1,3,3$NODES)*M(1,1,4$NODES))
 MI(1,1,3$NODES)=(M(1,1,2$NODES)*M(1,2,3$NODES)-M(1,2,2$NODES)*
$ M(1,1,3$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,1,3$NODES)=MI(1,3,1$NODES)-(M(1,1,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
 MI(1,1,3$NODES)=MI(1,3,1$NODES)+(M(1,1,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
 MI(1,1,3$NODES)=MI(1,3,1$NODES)+(M(1,2,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,1,3$NODES)=MI(1,3,1$NODES)-(M(1,2,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,1,3$NODES)=MI(1,3,1$NODES)+(M(1,4,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,4,3$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
 MI(1,2,3$NODES)=(M(1,1,1$NODES)*M(1,2,3$NODES)-M(1,2,1$NODES)*
$ M(1,1,3$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,2,3$NODES)=MI(1,3,2$NODES)-(M(1,1,1$NODES)*M(1,4,3$NODES)
$ -M(1,4,1$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
 MI(1,2,3$NODES)=MI(1,3,2$NODES)+(M(1,1,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
 MI(1,2,3$NODES)=MI(1,3,2$NODES)+(M(1,2,1$NODES)*M(1,4,3$NODES)
$ -M(1,4,1$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,2,3$NODES)=MI(1,3,2$NODES)-(M(1,2,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,2,3$NODES)=MI(1,3,2$NODES)+(M(1,4,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,4,3$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
 MI(1,3,3$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$ M(1,1,2$NODES))*(M(1,4,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,4,5$NODES))
 MI(1,3,3$NODES)=MI(1,3,3$NODES)-(M(1,1,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
```

(Continued)

Table D-3 (Continued)

```
 MI(1,3,3$NODES)=MI(1,3,3$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
 MI(1,3,3$NODES)=MI(1,3,3$NODES)+(M(1,2,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,3,3$NODES)=MI(1,3,3$NODES)-(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,3,3$NODES)=MI(1,3,3$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
 MI(1,4,3$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$ M(1,1,2$NODES))*(M(1,4,3$NODES)*M(1,5,5$NODES)-M(1,5,3$NODES)*
$ M(1,4,5$NODES))
 MI(1,4,3$NODES)=MI(1,3,4$NODES)-(M(1,1,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,2,5$NODES))
 MI(1,4,3$NODES)=MI(1,3,4$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,4,5$NODES)
$ -M(1,4,3$NODES)*M(1,2,5$NODES))
 MI(1,4,3$NODES)=MI(1,3,4$NODES)+(M(1,2,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,1,5$NODES))
 MI(1,4,3$NODES)=MI(1,3,4$NODES)-(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,4,5$NODES)
$ -M(1,4,3$NODES)*M(1,1,5$NODES))
 MI(1,4,3$NODES)=MI(1,3,4$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,1,3$NODES)*M(1,2,5$NODES)
$ -M(1,2,3$NODES)*M(1,1,5$NODES))
 MI(1,5,3$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$ M(1,1,2$NODES))*(M(1,4,3$NODES)*M(1,5,4$NODES)-M(1,5,3$NODES)*
$ M(1,4,4$NODES))
 MI(1,5,3$NODES)=MI(1,3,5$NODES)-(M(1,1,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,2,4$NODES))
 MI(1,5,3$NODES)=MI(1,3,5$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,4,4$NODES)
$ -M(1,4,3$NODES)*M(1,2,4$NODES)) .
 MI(1,5,3$NODES)=MI(1,3,5$NODES)+(M(1,2,1$NODES)*M(1,4,2$NODES)
$ -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,1,4$NODES))
 MI(1,5,3$NODES)=MI(1,3,5$NODES)-(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,4,4$NODES)
$ -M(1,4,3$NODES)*M(1,1,4$NODES))
 MI(1,5,3$NODES)=MI(1,3,5$NODES)+(M(1,4,1$NODES)*M(1,5,2$NODES)
```

(Continued)

Table D-3 (Continued)

```
$ -M(1,5,1$NODES)*M(1,4,2$NODES))*(M(1,1,3$NODES)*M(1,2,4$NODES)
$ -M(1,2,3$NODES)*M(1,1,4$NODES))
 MI(1,1,4$NODES)=(M(1,1,2$NODES)*M(1,2,3$NODES)-M(1,2,2$NODES)*
$ M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,3,5$NODES))
 MI(1,1,4$NODES)=MI(1,4,1$NODES)-(M(1,1,2$NODES)*M(1,3,3$NODES)
$ -M(1,3,2$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
 MI(1,1,4$NODES)=MI(1,4,1$NODES)+(M(1,1,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
 MI(1,1,2$NODES)=MI(1,4,1$NODES)+(M(1,2,2$NODES)*M(1,3,3$NODES)
$ -M(1,3,2$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,1,4$NODES)=MI(1,4,1$NODES)-(M(1,2,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,1,5$NODES))
 MI(1,1,4$NODES)=MI(1,4,1$NODES)+(M(1,3,2$NODES)*M(1,5,3$NODES)
$ -M(1,5,2$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
 MI(1,2,4$NODES)=(M(1,1,1$NODES)*M(1,2,3$NODES)-M(1,2,1$NODES)*
$ M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,3,5$NODES))
 MI(1,2,4$NODES)=MI(1,4,2$NODES)-(M(1,1,1$NODES)*M(1,3,3$NODES)
$ -M(1,3,1$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
 MI(1,2,4$NODES)=MI(1,4,2$NODES)+(M(1,1,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
 MI(1,2,4$NODES)=MI(1,4,2$NODES)+(M(1,2,1$NODES)*M(1,3,3$NODES)
$ -M(1,3,1$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
 MI(1,2,4$NODES)=MI(1,4,2$NODES)-(M(1,2,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,1,5$NODES))
 MI(1,2,4$NODES)=MI(1,4,2$NODES)+(M(1,3,1$NODES)*M(1,5,3$NODES)
$ -M(1,5,1$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
 MI(1,3,4$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$ M(1,1,2$NODES))*(M(1,3,4$NODES)*M(1,5,5$NODES)-M(1,5,4$NODES)*
$ M(1,3,5$NODES))
 MI(1,3,4$NODES)=MI(1,4,3$NODES)-(M(1,1,1$NODES)*M(1,3,2$NODES)
$ -M(1,3,1$NODES)*M(1,1,2$NODES))*(M(1,2,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,2,5$NODES))
 MI(1,3,4$NODES)=MI(1,4,3$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
```

(Continued)

Table D-3 (Continued)

```
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
  MI(1,3,4$NODES)=MI(1,4,3$NODES)+(M(1,2,1$NODES)*M(1,3,2$NODES)
$ -M(1,3,1$NODES)*M(1,2,2$NODES))*(M(1,1,4$NODES)*M(1,5,5$NODES)
$ -M(1,5,4$NODES)*M(1,1,5$NODES))
  MI(1,3,4$NODES)=MI(1,4,3$NODES)-(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,1,5$NODES))
  MI(1,3,4$NODES)=MI(1,4,3$NODES)+(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
  MI(1,4,4$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$ M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,5,5$NODES)-M(1,5,3$NODES)*
$ M(1,3,5$NODES))
  MI(1,4,4$NODES)=MI(1,4,4$NODES)-(M(1,1,1$NODES)*M(1,3,2$NODES)
$ -M(1,3,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,2,5$NODES))
  MI(1,4,4$NODES)=MI(1,4,4$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,3,5$NODES)
$ -M(1,3,3$NODES)*M(1,2,5$NODES))
  MI(1,4,4$NODES)=MI(1,4,4$NODES)+(M(1,2,1$NODES)*M(1,3,2$NODES)
$ -M(1,3,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,5,5$NODES)
$ -M(1,5,3$NODES)*M(1,1,5$NODES))
  MI(1,4,4$NODES)=MI(1,4,4$NODES)-(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,3,5$NODES)
$ -M(1,3,3$NODES)*M(1,1,5$NODES))
  MI(1,4,4$NODES)=MI(1,4,4$NODES)+(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,2,5$NODES)
$ -M(1,2,3$NODES)*M(1,1,5$NODES))
  MI(1,5,4$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$ M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,5,4$NODES)-M(1,5,3$NODES)*
$ M(1,3,4$NODES))
  MI(1,5,4$NODES)=MI(1,4,5$NODES)-(M(1,1,1$NODES)*M(1,3,2$NODES)
$ -M(1,3,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,2,4$NODES))
  MI(1,5,4$NODES)=MI(1,4,5$NODES)+(M(1,1,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,3,4$NODES)
$ -M(1,3,3$NODES)*M(1,2,4$NODES))
  MI(1,5,4$NODES)=MI(1,4,5$NODES)+(M(1,2,1$NODES)*M(1,3,2$NODES)
$ -M(1,3,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,5,4$NODES)
$ -M(1,5,3$NODES)*M(1,1,4$NODES))
  MI(1,5,4$NODES)=MI(1,4,5$NODES)-(M(1,2,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,3,4$NODES)
$ -M(1,3,3$NODES)*M(1,1,4$NODES))
```

(Continued)

Table D-3 (Continued)

```
 MI(1,5,4$NODES)=MI(1,4,5$NODES)+(M(1,3,1$NODES)*M(1,5,2$NODES)
$ -M(1,5,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,2,4$NODES)
$ -M(1,2,3$NODES)*M(1,1,4$NODES))
 MI(1,1,5$NODES)=(M(1,1,2$NODES)*M(1,2,3$NODES)-M(1,2,2$NODES)*
$ M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)-M(1,4,4$NODES)*
$ M(1,3,5$NODES))
 MI(1,1,5$NODES)=MI(1,5,1$NODES)-(M(1,1,2$NODES)*M(1,3,3$NODES)
$ -M(1,3,2$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
 MI(1,1,5$NODES)=MI(1,5,1$NODES)+(M(1,1,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
 MI(1,1,5$NODES)=MI(1,5,1$NODES)+(M(1,2,2$NODES)*M(1,3,3$NODES)
$ -M(1,3,2$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,1,5$NODES)=MI(1,5,1$NODES)-(M(1,2,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$ NODES)*M(1,1,5$NODES))
 MI(1,1,5$NODES)=MI(1,5,1$NODES)+(M(1,3,2$NODES)*M(1,4,3$NODES)
$ -M(1,4,2$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
 MI(1,2,5$NODES)=(M(1,1,1$NODES)*M(1,2,3$NODES)-M(1,2,1$NODES)*
$ M(1,1,3$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)-M(1,4,4$NODES)*
$ M(1,3,5$NODES))
 MI(1,2,5$NODES)=MI(1,5,2$NODES)-(M(1,1,1$NODES)*M(1,3,3$NODES)
$ -M(1,3,1$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
 MI(1,2,5$NODES)=MI(1,5,2$NODES)+(M(1,1,1$NODES)*M(1,4,3$NODES)
$ -M(1,4,1$NODES)*M(1,1,3$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,2,5$NODES))
 MI(1,2,5$NODES)=MI(1,5,2$NODES)+(M(1,2,1$NODES)*M(1,3,3$NODES)
$ -M(1,3,1$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,1,5$NODES))
 MI(1,2,5$NODES)=MI(1,5,2$NODES)-(M(1,2,1$NODES)*M(1,4,3$NODES)
$ -M(1,4,1$NODES)*M(1,2,3$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$ -M(1,3,4$NODES)*M(1,1,5$NODES))
 MI(1,2,5$NODES)=MI(1,5,2$NODES)+(M(1,3,1$NODES)*M(1,4,3$NODES)
$ - M(1,4,1$NODES)*M(1,3,3$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$ -M(1,2,4$NODES)*M(1,1,5$NODES))
 MI(1,3,5$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$ M(1,1,2$NODES))*(M(1,3,4$NODES)*M(1,4,5$NODES)-M(1,4,4$NODES)*
$ M(1,3,5$NODES))
 MI(1,3,5$NODES)=MI(1,5,3$NODES)-(M(1,1,1$NODES)*M(1,3,2$NODES)
$ -M(1,3,1$NODES)*M(1,1,2$NODES))*(M(1,2,4$NODES)*M(1,4,5$NODES)
$ -M(1,4,4$NODES)*M(1,2,5$NODES))
 MI(1,3,5$NODES)=MI(1,5,3$NODES)+(M(1,1,1$NODES)*M(1,4,2$NODES)
```

(Continued)

Table D-3 (Continued)

```
$  -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,2,4$NODES)*M(1,3,5$NODES)
$  -M(1,3,4$NODES)*M(1,2,5$NODES))
  MI(1,3,5$NODES)=MI(1,5,3$NODES)+(M(1,2,1$NODES)*M(1,3,2$NODES)
$  -M(1,3,1$NODES)*M(1,2,2$NODES))*(M(1,1,4$NODES)*M(1,4,5$NODES)
$  -M(1,4,4$NODES)*M(1,1,5$NODES))
  MI(1,3,5$NODES)=MI(1,5,3$NODES)-(M(1,2,1$NODES)*M(1,4,2$NODES)
$  -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,1,4$NODES)*M(1,3,5$NODES)
$  -M(1,3,4$NODES)*M(1,1,5$NODES))
  MI(1,3,K$NODES)=MI(1,5,3$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$  -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,1,4$NODES)*M(1,2,5$NODES)
$  -M(1,2,4$NODES)*M(1,1,5$NODES))
  MI(1,4,5$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$  M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,4,5$NODES)-M(1,4,3$NODES)*
$  M(1,3,5$NODES))
  MI(1,4,5$NODES)=MI(1,5,4$NODES)-(M(1,1,1$NODES)*M(1,3,2$NODES)
$  -M(1,3,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,4,5$NODES)
$  -M(1,4,3$NODES)*M(1,2,5$NODES))
  MI(1,4,5$NODES)=MI(1,5,4$NODES)+(M(1,1,1$NODES)*M(1,4,2$NODES)
$  -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,3,5$NODES)
$  -M(1,3,3$NODES)*M(1,2,5$NODES))
  MI(1,4,5$NODES)=MI(1,5,4$NODES)+(M(1,2,1$NODES)*M(1,3,2$NODES)
$  -M(1,3,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,4,5$NODES)
$  -M(1,4,3$NODES)*M(1,1,5$NODES))
  MI(1,4,5$NODES)=MI(1,5,4$NODES)-(M(1,2,1$NODES)*M(1,4,2$NODES)
$  -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,3,5$NODES)
$  -M(1,3,3$NODES)*M(1,1,5$NODES))
  MI(1,4,5$NODES)=MI(1,5,4$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
$  -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,2,5$NODES)
$  -M(1,2,3$NODES)*M(1,1,5$NODES))
  MI(1,5,5$NODES)=(M(1,1,1$NODES)*M(1,2,2$NODES)-M(1,2,1$NODES)*
$  M(1,1,2$NODES))*(M(1,3,3$NODES)*M(1,4,4$NODES)-M(1,4,3$NODES)*
$  M(1,3,4$NODES))
  MI(1,5,5$NODES)=MI(1,5,5$NODES)-(M(1,1,1$NODES)*M(1,3,2$NODES)
$  -M(1,3,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,4,4$NODES)
$  -M(1,4,3$NODES)*M(1,2,4$NODES))
  MI(1,5,5$NODES)=MI(1,5,5$NODES)+(M(1,1,1$NODES)*M(1,4,2$NODES)
$  -M(1,4,1$NODES)*M(1,1,2$NODES))*(M(1,2,3$NODES)*M(1,3,4$NODES)
$  -M(1,3,3$NODES)*M(1,2,4$NODES))
  MI(1,5,5$NODES)=MI(1,5,5$NODES)+(M(1,2,1$NODES)*M(1,3,2$NODES)
$  -M(1,3,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,4,4$NODES)
$  -M(1,4,3$NODES)*M(1,1,4$NODES))
  MI(1,5,5$NODES)=MI(1,5,5$NODES)-(M(1,2,1$NODES)*M(1,4,2$NODES)
$  -M(1,4,1$NODES)*M(1,2,2$NODES))*(M(1,1,3$NODES)*M(1,3,4$NODES)
$  -M(1,3,3$NODES)*M(1,1,4$NODES))
  MI(1,5,5$NODES)=MI(1,5,5$NODES)+(M(1,3,1$NODES)*M(1,4,2$NODES)
```

(Continued)

Table D-3 (Continued)

```
$ -M(1,4,1$NODES)*M(1,3,2$NODES))*(M(1,1,3$NODES)*M(1,2,4$NODES)
$ -M(1,2,3$NODES)*M(1,1,4$NODES))
 DET(1$NODES)=M(1,1,1$NODES)*MI(1,1,1$NODES)
$               +M(1,1,2$NODES)*MI(1,2,1$NODES)
$               +M(1,1,3$NODES)*MI(1,3,1$NODES)
$      4 1      +M(1,1,4$NODES)*MI(1,1,4$NODES)
$               +M(1,1,5$NODES)*MI(1,5,1$NODES)
 NM1=NODES-1
 NM2=NODES-2
 DO 50 I=1,5
 DO 50 J=1,5

   MI(1,I,J$NODES)=MI(1,I,J$NODES)/DET(1$NODES)
50 CONTINUE
   ITER=0
10 CONTINUE
   ITER=ITER+1
   DO 100 I=1,5
   UP(1,I)=(1.-W)*U(1,I)
   UP(1,I)=UP(1,I)-W*MI(1,I,1)*(N(2,1,1)*U(2,1)+N(2,1,2)*U(2,2)
  $ +N(2,1,3)*U(2,3)+N(2,1,4)*U(2,4)+N(2,1,5)*U(2,5))
   UP(1,I)=UP(1,I)-W*MI(1,I,2)*(N(2,2,1)*U(2,1)+N(2,2,2)*U(2,2)
  $ +N(2,2,3)*U(2,3)+N(2,2,4)*U(2,4)+N(2,2,5)*U(2,5))
   UP(1,I)=UP(1,I)-W*MI(1,I,3)*(N(2,3,1)*U(2,1)+N(2,3,2)*U(2,2)
  $ +N(2,3,3)*U(2,3)+N(2,3,4)*U(2,4)+N(2,3,5)*U(2,5))
   UP(1,I)=UP(1,I)-W*MI(1,I,4)*(N(2,4,1)*U(2,1)+N(2,4,2)*U(2,2)
  $ +N(2,4,3)*U(2,3)+N(2,4,4)*U(2,4)+N(2,4,5)*U(2,5))
   UP(1,I)=UP(1,I)-W*MI(1,I,5)*(N(2,5,1)*U(2,1)+N(2,5,2)*U(2,2)
  $ +N(2,5,3)*U(2,3)+N(2,5,4)*U(2,4)+N(2,5,5)*U(2,5))
   UP(1,I)=UP(1,I)+W*(MI(1,I,1)*D(1,1)+MI(1,I,2)*D(1,2)+
  $ MI(1,I,3)*D(1,3)+MI(1,I,4)*D(1,4)+MI(1,I,5)*D(1,5))
   UP(2,I$NM2)=(1.-W)*U(2,I$NM2)
   UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,1$NM2)*(L(1,1,1$NM2)*U(1,1$NM2)
  $ +L(1,1,2$NM2)*U(1,2$NM2)+L(1,1,3$NM2)*U(1,3$NM2)+L(1,1,4$NM2)*
  $ U(1,4$NM2)+L(1,1,5$NM2)*U(1,5$NM2))
   UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,1$NM2)*(N(3,1,1$NM2)*U(3,1$NM2)
  $ +N(3,1,2$NM2)*U(3,2$NM2)+N(3,1,3$NM2)*U(3,3$NM2)+N(3,1,4$NM2)*
  $ U(3,4$NM2)+N(3,1,5$NM2)*U(3,5$NM2))
   UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,2$NM2)*(L(1,2,1$NM2)*U(1,1$NM2)
  $ +L(1,2,2$NM2)*U(1,2$NM2)+L(1,2,3$NM2)*U(1,3$NM2)+L(1,2,4$NM2)*
  $ U(1,4$NM2)+L(1,2,5$NM2)*U(1,5$NM2))
   UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,2$NM2)*(N(3,2,1$NM2)*U(3,1$NM2)
  $ +N(3,2,2$NM2)*U(3,2$NM2)+N(3,2,3$NM2)*U(3,3$NM2)+N(3,2,4$NM2)*
  $ U(3,4$NM2)+N(3,2,5$NM2)*U(3,5$NM2))
   UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,3$NM2)*(L(1,3,1$NM2)*U(1,1$NM2)
```

(Continued)

Table D-3 (Continued)

```
     $ +L(1,3,2$NM2)*U(1,2$NM2)+L(1,3,3$NM2)*U(1,3$NM2)+L(1,3,4$NM2)*
     $ U(1,4$NM2)+L(1,3,5$NM2)*U(1,5$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,3$NM2)*(N(3,3,1$NM2)*U(3,1$NM2)
     $ +N(3,3,2$NM2)*U(3,2$NM2)+N(3,3,3$NM2)*U(3,3$NM2)+N(3,3,4$NM2)*
     $ U(3,4$NM2)+N(3,3,5$NM2)*U(3,5$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,4$NM2)*(L(1,4,1$NM2)*U(1,1$NM2)
     $ +L(1,4,2$NM2)*U(1,2$NM2)+L(1,4,3$NM2)*U(1,3$NM2)+L(1,4,4$NM2)*
     $ U(1,4$NM2)+L(1,4,5$NM2)*U(1,5$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,4$NM2)*(N(3,4,1$NM2)*U(3,1$NM2)
     $ +N(3,4,2$NM2)*U(3,2$NM2)+N(3,4,3$NM2)*U(3,3$NM2)+N(3,4,4$NM2)*
     $ U(3,4$NM2)+N(3,4,5$NM2)*U(3,5$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,5$NM2)*(L(1,5,1$NM2)*U(1,1$NM2)
     $ +L(1,5,2$NM2)*U(1,2$NM2)+L(1,5,3$NM2)*U(1,3$NM2)+L(1,5,4$NM2)*
     $ U(1,4$NM2)+L(1,5,5$NM2)*U(1,5$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)-W*MI(2,I,5$NM2)*(N(3,5,1$NM2)*U(3,1$NM2)
     $ +N(3,5,2$NM2)*U(3,2$NM2)+N(3,5,3$NM2)*U(3,3$NM2)+N(3,5,4$NM2)*
     $ U(3,4$NM2)+N(3,5,5$NM2)*U(3,5$NM2))
      UP(2,I$NM2)=UP(2,I$NM2)+W*(MI(2,I,1$NM2)*D(2,1$NM2)+MI(2,I,2$NM2)
     $ *D(2,2$NM2)+MI(2,I,3$NM2)*D(2,3$NM2)+MI(2,I,4$NM2)*D(2,4$NM2)
     $ +MI(2,I,5$NM2)*D(2,5$NM2))
      UP(NODES,I)=(1.-W)*U(NODES,I)
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,1)*(L(NM1    ,1,1)*U(NM1,1)
     $ +L(NM1    ,1,2)*U(NM1,2)+L(NM1    ,1,3)*U(NM1,3)+L(NM1    ,1,4)
     $ *U(NM1,4)+L(NM1    ,1,5)*U(NM1,5))
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,2)*(L(NM1    ,2,1)*U(NM1,1)

     $ +L(NM1    ,2,2)*U(NM1,2)+L(NM1    ,2,3)*U(NM1,3)+L(NM1    ,2,4)
     $ *U(NM1,4)+L(NM1    ,2,5)*U(NM1,5))
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,3)*(L(NM1    ,3,1)*U(NM1,1)
     $ +L(NM1    ,3,2)*U(NM1,2)+L(NM1    ,3,3)*U(NM1,3)+L(NM1    ,3,4)
     $ *U(NM1,4)+L(NM1    ,3,5)*U(NM1,5))
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,4)*(L(NM1    ,4,1)*U(NM1,1)
     $ +L(NM1    ,4,2)*U(NM1,2)+L(NM1    ,4,3)*U(NM1,3)+L(NM1    ,4,4)
     $ *U(NM1,4)+L(NM1    ,4,5)*U(NM1,5))
      UP(NODES,I)=UP(NODES,I)-W*MI(NODES,I,5)*(L(NM1    ,5,1)*U(NM1,1)
     $ +L(NM1    ,5,2)*U(NM1,2)+L(NM1    ,5,3)*U(NM1,3)+L(NM1    ,5,4)
     $ *U(NM1,4)+L(NM1    ,5,5)*U(NM1,5))
      UP(NODES,I)=UP(NODES,I)+W*(MI(NODES,I,1)*D(NODES,1)+MI(NODES,I,2)
     $ *D(NODES,2)+MI(NODES,I,3)*D(NODES,3)+MI(NODES,I,4)*D(NODES,4)
     $ +MI(NODES,I,5)*D(NODES,5))
  100 CONTINUE
      DO 350  I=1,5
      DIF(1,I$NODES)= UP(1,I$NODES)-U(1,I$NODES)
```

(Continued)

Table D-3 (Concluded)

```
350 CONTINUE
    RMS=0.
    DEL=0.
    DO 360 I=1,5
    DO 360 J=1,NODES
    DEL=DEL+DIF(J,I)*DIF(J,I)
    RMS=RMS+UP(J,I)*UP(J,I)
360 CONTINUE
    DEL=SQRT(DEL)
    RMS=SQRT(RMS)
    TEST=DEL/RMS
    WRITE(6,450)UP,J,DIF
450 FORMAT(9E10.3)
    WRITE(6,500) ITER,TEST,DEL,RMS
500 FORMAT(I5,3E10.3)
    DO 400 I=1,5
    U(1,I$NODES)=UP(1,I$NODES)
400 CONTINUE
    IF(TEST.LE.EPS.OR.ITER.GE.MAXI)RETURN
    GO TO 10
    END
```

1. Report No. NASA CR-3369	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle HYPERBOLIC/PARABOLIC DEVELOPMENT FOR THE GIM-STAR CODE		5. Report Date December 1980
		6. Performing Organization Code
7. Author(s) L. W. Spradley, J. F. Stalnaker, A. W. Ratliff		8. Performing Organization Report No. LMSC-HREC TR D697882
		10. Work Unit No.
9. Performing Organization Name and Address Lockheed Missiles & Space Company, Inc. Huntsville Research & Engineering Center 4800 Bradford Blvd. Huntsville, AL 35807		11. Contract or Grant No. NAS1-15783, NAS1-15795
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitors: James L. Hunt and J. Phillip Drummond.
Progress Report

Appendix A by L. W. Spradley, Lockheed Missiles & Space Company, Inc.
Appendix B by Jürgen Thoenes, Lockheed Missiles & Space Company, Inc.
Appendix C by John F. Stalnaker, Lockheed Missiles & Space Company, Inc.
Appendix D by S. J. Robertson, Lockheed Missiles & Space Company, Inc.

16. Abstract

A study was conducted to compute flow fields in supersonic inlet configurations. The problem was run with the elliptic GIM code on the STAR computer. Spillage flow under the lower cowl was calculated to be 33% of the incoming stream. The shock/boundary layer interaction on the upper propulsive surface was computed including separation. All shocks produced by the flow system were captured. An investigation of linearized block implicit (LBI) schemes was also conducted to determine their application to the GIM code. Pure explicit methods have stability limitations and fully implicit schemes are inherently inefficient; however, LBI schemes show promise as an effective compromise. This study also included the development of a quasi-parabolic version of the GIM code. The basic idea was to make use of classical parabolized Navier-Stokes methods combined with quasi-time relaxation. This scheme is referred to as quasi-parabolic although it applies equally well to hyperbolic supersonic inviscid flows. Second order windward differences are used in the marching coordinate and either explicit or linear block implicit time relaxation can be incorporated.

17. Key Words (Suggested by Author(s)) Inlet flows Shock/boundary layer interaction Linearized block implicit schemes Parabolized Navier-Stokes	18. Distribution Statement Unclassified — Unlimited Subject Category 34

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 175	22. Price A08

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1980