Loughborough University

A NOVEL FINITE ELEMENT TECHNIQUE FOR THE SOLUTION OF ENGINEERING FLOW PROBLEMS

By

Aroba Khan-Siddiqui

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy of Loughborough University.

2003-2007

Department of Chemical Engineering

Advanced Separation Technologies Research Group

Acknowledgements

I would like to express my sincere gratitude to my supervisors Professor Vahid Nassehi and Professor Richard Wakeman for their guidance, advice and support during this research. I am equally thankful to Engineering and Physical Research Council, UK for the financial support which has made this work possible.

I am greatly indebted to my parents, brother, sisters and nieces for the love and support they gave me enabling me to get through my studies. The patience and encouragement of my dear husband Shahid during this work was very much appreciated. Contributions from my friends in terms of moral support, backing and assistance was invaluable.

It should not go unmentioned that the people whom I have interacted with in this department such as Yasmin Kosar and Paul Izzard have been of great help.

To My Dear Dad

Abstract

A new technique known as the bubble function method is developed for the modelling of fluid flow problems. The main motivation for this work has been the desire to resolve difficulties that traditional methods show in dealing with multi-scale behaviour in flow regimes. All of the traditionally used methods require excessive mesh refinement in the simulations of systems that combine different scale of behaviour in one domain. The present bubble function method avoids such crude remedies and instead of using an elegant mathematical technique for the conjunctive approximation of fine and coarse scale phenomena. Using numerical experiments it is shown that the implementation of the bubble function method generates accurate and stable solutions for a wide range of problems. This range includes convection and reaction dominated transport phenomena, and various types of porous flow systems, it can also be extended to transient flow simulations. To demonstrate the applicability of the present technique it has been used to solve a realistic problem, namely solute dispersion in an estuary. The results of this simulation show good agreement with field survey data.

Table of Contents

List of Figures

Nomenclature

Symbols	Definition
N_i	Elemental shape function
W_i	Weight function
p	Pressure, Pa
P	Normalised Pressure
u	Velocity in the x direction, ms^{-1}
v	Velocity in the y direction, ms^{-1}
c	Heat capacity, $Jg^{-1}K^{-1}$
t	Time variable, s

Greek Letters

μ	Fluid viscosity, $kgms^{-1}$
μ_e	Effective fluid viscosity, $kgms^{-1}$
Ω	Solution domain
Ω_e	Element domain
ρ	Fluid density, kgm^{-3}
Γ	Boundary of solution domain
Γ_e	Elemental Boundary
θ	Time stepping parameter

Subscripts

e	Represents elemental domain
x	Indicates the x-direction
y	Indicates the y-direction

Chapter 1
Introduction

1.1 Overview

Existing techniques used in the modelling of field problems in engineering cannot cope with the difficulties arising in many fluid flow simulations which result from the multi-scale behaviour of such regimes. Therefore schemes using finite difference, finite volume and finite element methods require modifications to be applicable to systems that combine different scales of behaviour in one domain. Traditionally these methods have relied on the use of excessive mesh refinement to deal with multi-scale problems. Although such remedies solve problems related to multi-scale behaviour they use excessive computational power and time, and hence are not practical.

The research discussed in the following chapters is the extension of the newly emerging bubble function method to solve multi-scale problems arising in practical flow problems. In this work the use of bubble function method in conjunction with finite element schemes has been extended to develop solutions for convection dominated problems, transient diffusion problems and porous flow problems. The method has also been used to solve a typical realistic flow problem and compare its output with experimental results.

1.2 Aims and Objectives

The principle aim of this research is to develop a simulation scheme for multi-scale fluid flow problems using the bubble function method as a practical numerical technique. The objectives of this project have been

- To develop a fast and efficient scheme to simulate multi-scale flow regimes.

- To validate the developed scheme under various scenarios. In particular evaluate the performance of the scheme by comparison of its results with analytical results, results obtained using other numerical methods and experimental data.

1.3 Thesis Structure

The scope of the thesis is outlined as follows

Chapter 2 deals with the mathematical background that has led to the development of the bubble function approach. The techniques is described, step by step, via the use of examples to avoid excessive mathematical formalism that would have been required to explain the techniques of the bubble function approach from a theoretical point of view. This chapter also gives a physical description of the concept of multi-scale behaviour in fluid flow regimes.

Chapter 3 illustrates the basic approach used in the bubble function method via the solution of one dimensional differential equation. This discussion is then related to compare the method with other finite element schemes that have been previously applied. Following this, the chapter concentrates on the extension of the bubble function method to two and three dimensional problems.

Chapter 4 presents sample simulations showing the applicability and performance of the bubble function method in dealing with practical problems. Including comparisons of the simulated results obtained using the bubble function method with simulations generated by other finite element schemes and analytical results as well as experimental data.

Chapter 5 in this chapter the conclusion shown from this research and further suggestions for future research have been discussed.

To enhance the readability of the thesis, instead of giving a full description of all of the published literature in one chapter, I have cited and discussed the literature in corresponding sections of the thesis for each topic.

Chapter 2
Theoretical Background

2.1 Introduction

In this chapter the mathematical background that has led to the development of the bubble function approach is outlined. To avoid lengthy mathematical discussions these ideas are presented via examples. Analytical solutions which provide a basis for construction of higher order approximation for engineering flow problems are presented and the discussion extended to the incorporation of such solutions in discretised domains.

This chapter also includes a brief discussion regarding the physical nature of typical multi-scale behaviour that makes the use of higher order approximations desirable. Combination of ideas of discretised solutions with higher order approximations and features of multi-scale behaviour is the under pinning basis for the development of bubble enriched finite element solutions.

2.2 Analytical Solution of a typical Convection-Diffusion Problem

We start with the analytical solution of a two dimensional convection-diffusion problem given in terms of the following P.D.E

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}-k_1\frac{\partial\phi}{\partial x}-k_2\frac{\partial\phi}{\partial y}=0 \qquad \textbf{(2.1)}$$

where $\phi(x\,,\,y) \in \Re = [0,1] \times [0,1]$

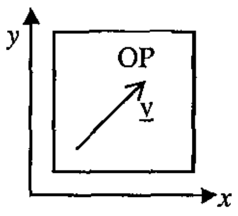
and $k_1, k_2 >> 0$

Equation (2.1) should be solved subject to the boundary conditions of :

$$\phi(0, y) = \phi(x, 0) = 1 \ (x , y \neq 1)$$

$$\phi(1, y) = \phi(x, 1) = 0 \ (x , y \neq 0)$$

$$\phi(0, 1) = \phi(1, 0) = ½$$



$\underline{v}$ is in this direction for all k_1, $k_2 > 0$

Suppose that

$$\phi(x, y) = X(x)Y(y) \tag{2.2}$$

Substituting equation (2.2) into equation (2.1) gives

$$\frac{X''-k_1X'}{X} = -\frac{Y''+k_2Y'}{Y} \tag{2.3}$$

which gives

$$X''-k_1X'-MX = 0 \tag{2.4}$$

$$Y''-k_2Y'-MY = 0 \tag{2.5}$$

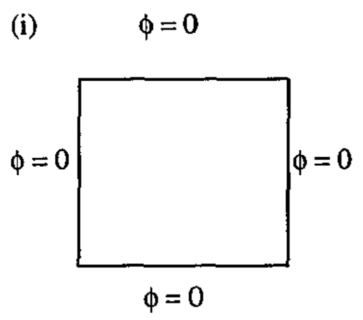
where M is a constant value

These ordinary differential equations give the following

$$X(x)=e^{\frac{k_1x}{2}}\left[Ae^{\sqrt{k_1^2+4M}\,x/2}+Be^{-\sqrt{k_1^2+4M}\,x/2}\right] \tag{2.6}$$

$$Y(y)=e^{\frac{k_2y}{2}}\left[Ce^{\sqrt{k_2^2+4M}\,y/2}+De^{-\sqrt{k_2^2+4M}\,y/2}\right] \tag{2.7}$$

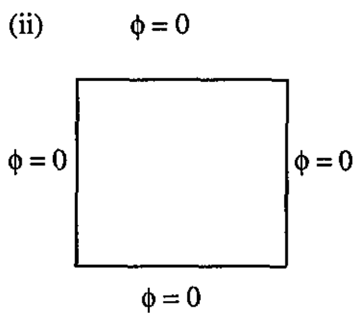
(i)



$\phi(x, 0) = 1 \quad 0 < x < 1$

$\phi(x, 1) = \phi(0, y) = \phi(1, y) = 0$

(ii)



$\phi(x, 0) = 1 \quad 0 < x < 1$

$\phi(x, 0) = \phi(x, 1) = \phi(1, y) = 0$

The solution of (2.1) excluding the points (0,0), (0,1) (1,0) which are dealt with separately is given as the sum of the solutions of (i) and (ii).

Consider case (i) first – we have

$$X(0)=X(1)=0 \tag{2.8}$$

The form of the solution in equation (2.6) is dictated by the sign of k_1^2+4M

If k_1^2+4M is positive we have

$$X(x)=e^{\frac{k_1x}{2}}\left[(A+B)\cosh\frac{\sqrt{k_1^2+4M}}{2}x+(A-B)\sinh\frac{\sqrt{k_1^2+4M}}{2}x\right]=0 \tag{2.9}$$

from equation (2.8) X(0) = 0 gives A+B=0 and X(1) = 0 gives $\sinh \frac{\sqrt{k_1^2+4M}}{2} x = 0$ in this case

A–B=0 for non-trivial solution therefore in this case it is impossible to give any real answers.

If $k_1^2 + 4M$ is negative we have from equation (2.6)

$$X(x) = e^{\frac{k_1 x}{2}} \left[A e^{i\sqrt{-k_1^2-4M}\,x/2} + B e^{-i\sqrt{-k_1^2-4M}\,x/2} \right] \qquad \textbf{(2.10)}$$

or

$$X(x) = e^{\frac{k_1 x}{2}} \left[(A+B)\cos\left(\sqrt{-k_1^2-4M}\; x/2\right) + (A-B)i\sin\left(\sqrt{-k_1^2-4M}\; x/2\right) \right] \qquad \textbf{(2.11)}$$

From X(0) = 0, A+B=0 and X(1)=0 gives

$$Sin\left(\sqrt{-k_1^2-4M}\; x/2\right) = 0$$

or $\frac{1}{2}\sqrt{-k_1^2-4M} = n\Pi \qquad n = 1, 2, \ldots \qquad$ **(2.12)**

i.e. $M = -\frac{1}{4}\sqrt{k_1^2 - 4n^2\Pi^2}$

and we have

$$X(x) = (A-B)ie^{\frac{k_1 x}{2}} \sin n\Pi x \qquad \textbf{(2.13)}$$

and $k_1^2 + 4M < 0$ is the only possible case. We define

$$An = k_2^2 - 4M = k_1^2 - k_2^2 + 4n^2\Pi^2 > 0 \qquad \textbf{(2.14)}$$

Similarly for constants C' and D' equation (2.7) gives

$$Y(y) = e^{\frac{k_2 y}{2}} \left[C' \cosh\left(\sqrt{An} \, (1-y) \big/ 2 \right) + D' \sinh\left(\sqrt{An} \, (1-y) \big/ 2 \right) \right] \qquad \textbf{(2.15)}$$

Using Y(1) = 0 gives C' = 0 thus

$$Y(y) = D' e^{\frac{k_2 y}{2}} \sinh\left(\sqrt{An} \, (1-y) \big/ 2 \right) \qquad \textbf{(2.16)}$$

We do not require any condition on Y(0) since Y(0) = 0 ensures $\phi(0,0) = 0$. Thus for constant α

$$\phi_1(x,y) = \alpha e^{\frac{(k_1 x + k_2 y)}{2}} \sin \Pi x \sinh\left(\sqrt{An} \, (1-y) \big/ 2 \right) \qquad \textbf{(2.17)}$$

The most general solution for boundary conditions given by (2.1) is a linear combination of the particular solution i.e.

$$\phi_1(x,y) = \sum_{n=1}^{\infty} C_n e^{(k_1 x + k_2 y)/2} \sin \Pi x \sinh\left(\sqrt{An} \, (1-y) \big/ 2 \right) \qquad \textbf{(2.18)}$$

where the coefficients C_n (n=1,2,...) are determined from $\phi(x,0) = 1$ using orthogonal functions and Fourier Series techniques.

Putting y=0 and $b_n = C_n \sinh\left(\sqrt{An}/2\right)$ we have

$$\phi_1(x,0) = \sum_{n=1}^{\infty} b_n e^{\frac{k_1 x}{2}} \sin \Pi x = 1 \qquad \textbf{(2.19)}$$

functions $e^{\frac{k_1x}{2}}\sin n\Pi x$ and $e^{\frac{k_1x}{2}}\sin m\Pi x$ are orthogonal with respect to e^{-k_1x} (positive weighting). Thus simplifying equation (2.19) by $e^{\frac{k_1x}{2}}\sin m\Pi x$, weighting by e^{-k_1x} and integrating over intervals 0→1 gives

$$\sum b_n\int_0^1 e^{-k_1x}e^{\frac{k_1x}{2}}\sin \Pi x e^{\frac{k_1x}{2}}\sin m\Pi x dx=\int_0^1 1.e^{-k_1x}e^{\frac{k_1x}{2}}\sin m\Pi x dx \qquad \textbf{(2.20)}$$

this term on the left hand side of equation (2.20) except for m=n.

And

$$b_n\int_0^1 \sin^2 n\Pi x dx=\int_0^1 e^{-\frac{k_1x}{2}}\sin n\Pi x dx \qquad \textbf{(2.21)}$$

$$b_n=\frac{\int_0^1 e^{-\frac{k_1x}{2}}\sin n\Pi x dx}{\int_0^1 \sin^2 n\Pi x dx}=\frac{8n\Pi\left[1-(-1)^n e^{-k_1/2}\right]}{k_1^2+4n^2\Pi^2} \qquad \textbf{(2.22)}$$

and equation (2.18) becomes

$$\phi_1(x,y)=\sum_{n=1}^{\infty}\frac{8n\Pi\left[1-(-1)^n e^{-k_1/2}\right]e^{(k_1x+k_2y)/2}\sin \Pi x\sinh\left(\sqrt{An}\,{(1-y)}\!\big/_{2}\right)}{k_1^2+4n^2\Pi^2\sinh\left(\sqrt{An}/2\right)} \qquad \textbf{(2.23)}$$

in an identical manner we obtain $\phi_2(x,y)$ the solution for case (ii). The general solution of equation (2.1) is then the sum of ϕ_1 and ϕ_2 plus the values of ϕ at the points (0,0), (0,1) and (1,0) and is given by

$\phi(0,0) = 1$, $\phi(0,1) = \phi(1,0) = ½$

$$\phi(x,y)=\sum_{n=1}^{\infty}\frac{8n\Pi e^{(k_1x+k_2y)/2}}{\sinh\left(\sqrt{An}/2\right)}\left[\begin{array}{l}\left(\frac{\left[1-(-1)^n e^{-k_1/2}\right]}{k_1^2+4n^2\Pi^2}\right)\sin\Pi x\sinh\left(\sqrt{An}\,{}^{(1-y)}\!/_{2}\right)+\\ \left(\frac{\left[1-(-1)^n e^{-k_2/2}\right]}{k_1^2+4n^2\Pi^2}\right)\sin\Pi y\sinh\left(\sqrt{An}\,{}^{(1-x)}\!/_{2}\right)\end{array}\right] \quad \textbf{(2.24)}$$

Boundary conditions at points $\phi(0,0) = 1$, $\phi(0,1) = \phi(1,0) = ½$ must be fixed and k_1 is small

$$\sum_{1}^{\infty}\frac{f_n\frac{n\Pi}{2}}{\frac{n\Pi}{2}}e^{\frac{k_1}{4}}\left(1+e^{-\frac{k_1}{2}}\right)$$

$$\phi(x,0)=\sum_{n=1}^{\infty}\frac{8n\Pi e^{\frac{k_1x}{2}}\left[1-(-1)^n e^{-\frac{k_1}{2}}\right]\sin\Pi x}{k_1^2+4n^2\Pi^2},\ \phi(0,y)=\sum_{n=1}^{\infty}\frac{8n\Pi e^{\frac{k_2y}{2}}\left[1-(-1)^n e^{-\frac{k_2}{2}}\right]\sin\Pi y}{k_2^2+4n^2\Pi^2}$$

$$\sum_{n=1}^{\infty}\frac{n\sin\Pi x}{k_2^2+4n^2\Pi^2} \quad \textbf{(2.25)}$$

If the PDE has a term such as Q included in the RHS of equation (2.1) a complete solution will require the addition of the below source term to equation (2.25).

$$-\frac{1}{2}\left(\frac{x}{k_1}+\frac{y}{k_2}\right)Q$$

2.3 Solution of the Convection-Diffusion equation using 'Domain Discretisation' techniques

A different solution for the problem represented by equation (2.1) can be obtained using a domain discretisation. This has the advantage that it can be used to obtain solutions under more general boundary conditions than the specific conditions that were used to generate the previous results. The solution can also be extended to irregular domains. However we will use a square domain and boundary conditions similar to those given previously to be able to directly compare these solutions with the analytical results. We will also explore the affects of increasing the coefficient of the first order term (i.e. convection term) and character of the solutions that can be obtained using smoothing (i.e. upwinding) techniques.

2.3.1 One Dimensional Case

We start with a one dimensional convection diffusion problem given as

$$\frac{d^2\phi(x)}{dx^2} - K(x)\frac{d\phi(x)}{dx} = S(x) \text{in domain } x_1 \leq \Omega \leq x_2 \tag{2.26}$$

Equation (2.25) is subject to essential boundary conditions of $x = x_1$, $\phi = \phi(a)$ and $x = x_2$, $\phi = \phi(b)$. In order to develop a weighted residual finite element solution for equation (2.25) the domain Ω is discretised into a mesh of finite elements. Within every element (Ω_e) a weak variational form of equation (2.25) is derived by integrating the functional which results from replacing $\phi(x)$ by a trial function $\phi^h(x)$ and weighting the general residual:

$$\int_{\Omega_e} w(x)\left\{\frac{d^2\phi^h}{dx^2} - K(x)\frac{d\phi^h}{dx} - S(x)\right\}d\Omega_e = 0 \tag{2.27}$$

where w(x) is a weight function and

$$\phi^h = \sum_{i=1}^{m} N_i\phi_i \tag{2.28}$$

N_i is the interpolation function associated with node i; m is the number of nodes per element and ϕ_i represents the nodal value of ϕ. In the standard Bubnov-Galerkin method the weight function is taken to be identical to the interpolation function ($w_j = N_i$ for i = j). Integration of equation (2) by parts gives (note that in one-dimensional case $d\Omega_e$ is simply dx).

$$-\int_{\Omega_e} \frac{dw}{dx}\frac{d\phi^h}{dx}dx - \int_{\Omega_e} w.K\frac{d\phi^h}{dx}dx - \int_{\Omega_e} w.S.dx + w\frac{d\phi^h}{dx}\bigg|_{\Omega_e} = 0 \qquad \textbf{(2.29)}$$

Thus by analogy to matrix forms with summation over the repeated index i, the basic weighted residual finite element form of the original convection diffusion equation becomes:

$$\left[\int_{\Omega_e} \frac{dN_j}{dx}\frac{dN_i}{dx}dx - \int_{\Omega_e} K.N_j\frac{dN_i}{dx}dx - \int_{\Omega_e} S.N_j dx\right]\{\Phi_i\} = \left\{N_j \frac{d\sum N_i\phi_i}{dx}\bigg|_{\Omega_e}\right\} \qquad \textbf{(2.30)}$$

i,j= 1, 2,---------------m

Using an isoperimetric mapping of the form

$$x = \sum_{i=1}^{m} N_i(\xi)x_i \qquad \textbf{(2.31)}$$

Equation (2.29) is cast in a local natural coordinate system for a master element defined between ξ = -1 and ξ = +1 and the integrals in its left hand side are evaluated by Gaussian quardrature. This process is repeated for every element and finally all of the resulting elemental equations are assembled together (Zienkiewicz 1977). The flux term in the right hand side of equation (2.29) vanishes for all inter-element boundaries and appears only on the exterior boundaries of the solution domain. Application of the boundary conditions renders the assembled global set of equations determinate and solvable. However, if the coefficient of the first order derivative (K) in equation (2.25) is large (convection dominated case) and the selected interpolation function (N_i) are polynomials (or for the multi-dimensional situations are the tensor products of polynomials) the solution of the obtained global set will yield oscillatory and unreliable results. In particular if linear interpolation functions are used the described Bubnov-Galerkin scheme will produce an oscillatory solution, which is, identical to the one obtained by a finite difference technique based on central differences. In the finite difference context the traditional way of solving this problem is to use less accurate forward (or backward) for the first order derivative term (Roache 1972).

The stream-lined upwinding Petrov-Galerkin modification of the described finite element procedure presented by Brooks and Hughes is based on using a weighting function which is given by:

$$W_i(\xi) = N_i(\xi) + \left[\coth\frac{Kh}{2} - \frac{2}{Kh}\right]\frac{\partial N_i(\xi)}{\partial \xi} \qquad \textbf{(2.32)}$$

where h is the element length. They have shown that using this weighting a nodal exact solution for the original equation can be obtained. Such a rigorous analysis is not possible for two or three dimensional problem and an analogous form for weight function in tow dimensions is given by (Nassehi, 2002)

$$W_i(\xi,\eta) = N_i(\xi,\eta) + K.d.\frac{\partial N_i(\xi,\eta)}{\partial \xi} + K.d.\frac{\partial N_i(\xi,\eta)}{\partial \eta} \qquad \textbf{(2.33)}$$

Where d is an upwinding constant which is a function of the so called nominal element length. The stream-lining concept arises from the idea of trying to define the upwinding constant d in a way which eliminates spurious diffusions in the crosswind direction.

A different equation of the form similar to equation (2.25) will have a solution consisting of two components-the particular integral and the complementary function (corresponding to the solution when S=O). The exponential interpolating functions are really designed to model the complementary function. In one dimension we discretise the solution domain into a mesh of bi-nodal elements of length h_ξ. In terms of ξ (i.e. the variable along the corresponding master element) we define the following interpolation functions:

$$N_{+1} = \frac{e^{Kh_\xi(\xi+1)/2} - 1}{e^{Kh_\xi} - 1}$$

$$N_{-1} = \frac{e^{Kh_\xi} - e^{Kh_\xi(\xi+1)/2}}{e^{Kh_\xi} - 1}$$

Firstly, we note that these functions have correct local support i.e.

$N_{+1}(+1) = 1$ and $N_{+1}(-1) = 0$

$N_{-1}(-1) = 1$ and $N_{-1}(+1) = 0$

Secondly, they are square integrable (L_2) functions satisfying the necessary continuity, differentiability and smoothness, required for the finite element solution of equation (2.1). If we transform from our master element back to the global system we could write these interpolation function as:

$$N_{+1} = \frac{e^{Kx} - 1}{e^{Kh} - 1} \qquad 0 \le x \le h$$
$$N_{-1} = \frac{e^{Kh} - e^{Kx}}{e^{Kh} - 1}$$

It is easy then to see that these interpolation functions satisfy the homogeneous form of the equation (2.25) exactly if K is constant.

We now extend the above solution to a two dimensional domain, we consider the two dimensional convection-diffusion equation represented by

$$\frac{\partial^2\phi(x,y)}{\partial x^2}+\frac{\partial^2\phi(x,y)}{\partial y^2}-K_1\frac{\partial\phi(x,y)}{\partial x}-K_2\frac{\partial\phi(x,y)}{\partial y}=0 \qquad \textbf{(2.34)}$$

2.3.2 Two Dimensional Case

In two dimensional smooth domain Ω with a closed boundary Γ. Equation (2.33) should be solved using finite element method subject to general essential boundary conditions, including those that were used in obtaining the analytical solution previously in 2.1.1 we can obtain the solution for the same problem and compare the results (for the case when **K₁** and **K₂** are constants) by the separation of variables.

In order to formulate a finite element solution for equation (2.33) using exponential functions we construct a set of tensor product elements based on the one dimensional exponential shape functions. For a four nodded master element we have

$$\begin{Bmatrix} M_1(\xi,\eta)=N(\xi_{-1}).N(\eta_{-1}) \\ M_2(\xi,\eta)=N(\xi_{+1}).N(\eta_{-1}) \\ M_3(\xi,\eta)=N(\xi_{+1}).N(\eta_{+1}) \\ M_4(\xi,\eta)=N(\xi_{-1}).N(\eta_{+1}) \end{Bmatrix} \qquad \textbf{(2.35)}$$

Where $N(\xi_{-1})$ and $N(\xi_{+1})$ are given in equation (2.33) we get $N(\eta_{-1})$ and $N(\eta_{+1})$. For Bubnov-Galerkin formulations we use exponential weight functions which are identical to interpolation functions. For Petrov-Galerkin (Upwinding) formulations we use modified weight functions given either by

$$W_i = M_i + \frac{K_1^2}{K_1^2+K_2^2}\left(\coth\frac{K_1 h_\xi}{2} - \frac{2}{K_1 h_\xi}\right)\frac{\partial M_i}{\partial \xi}$$

$$+\frac{K_2^2}{K_1^2+K_2^2}\left(\coth\frac{K_2 h_\xi}{2} - \frac{2}{K_2 h_\xi}\right)\frac{\partial M_i}{\partial \eta} \tag{2.36}$$

(we refer to this scheme as exponential upwinding scheme A); or by

$$W_i = M_i + \frac{K_1 h_\xi}{2\sqrt{K_1^2+K_2^2}}\frac{\partial M_i}{\partial \xi} + \frac{K_2 h_\eta}{2\sqrt{K_1^2+K_2^2}}\frac{\partial M_i}{\partial \eta} \tag{2.37}$$

(we refer to this scheme as exponential upwinding scheme B).

In this two dimensional problem K_1 or K_2 (coefficients of the first order derivatives) are constants and it is possible to evaluate the integrals in the elemental stiffness equation analytically. This is used to derive the working equation of the present solution scheme. We consider the results of various values of K_1 and K_2.

Starting with $K_1 = K_2 = 2.5$ to 10. we obtain first set of results for the case where convection terms are comparatively small. Results for a 7x7 finite element mesh (Figure 2.2) for various cases are given in figure 2.3. Bubnov-Galerkin schemes give acceptable results although for $K_1 = K_2 = 10$ those solutions which are based on linear interpolation function start to oscillate. Upwinded schemes in general produce over damped solution. However the deviation from the exact solution is slight for consistent Petrov-Galerkin based on exponential functions and sever for bi-quadratic functions.

Increasing $K_1 = K_2$ to 40-160 differences arising from various approximating functions emerge. The exact solution tends to be 1.0 for all interior nodes. The Bubnov—Galerkin formulations based on bilinear and bi-quadratic functions produces oscillatory and useless solutions. Upwinded Petrov-Galerkin schemes based on polynomial functions produce over damped solutions. Upwinded Petrov-Galerkin schemes A and B based on exponential functions produce accurate results with slight oscillations. The Bubnov-Galerkin scheme based on exponential functions produce the best results. These are shown in figure 2.3.

As K_1 and K_2 increase beyond 160 (i.e. the convection terms become more dominant) the upwinding schemes (based on polynomial or exponential functions) become less effective. In fact they tend to produce very nearly the same nodal values irrespective of K_1 and K_2 values. In contrast the Bubnov-Galerkin scheme based on exponential functions produces more and more accurate results as K_1 and K_2 increase. With $K = 90000$ these results are accurate to 6 decimal places, (Nassehi and King, 1991).

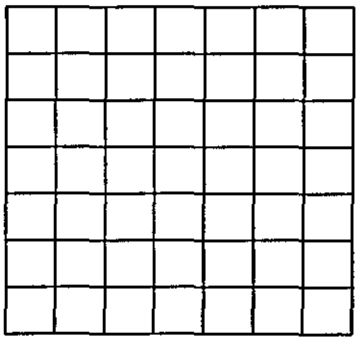


Figure 2.1 A Simple Domain Discretisation

[The simple domain discretisation shown in figure 2.1 makes the use of tensor product of exponentials as the shape functions in the finite element scheme possible.]

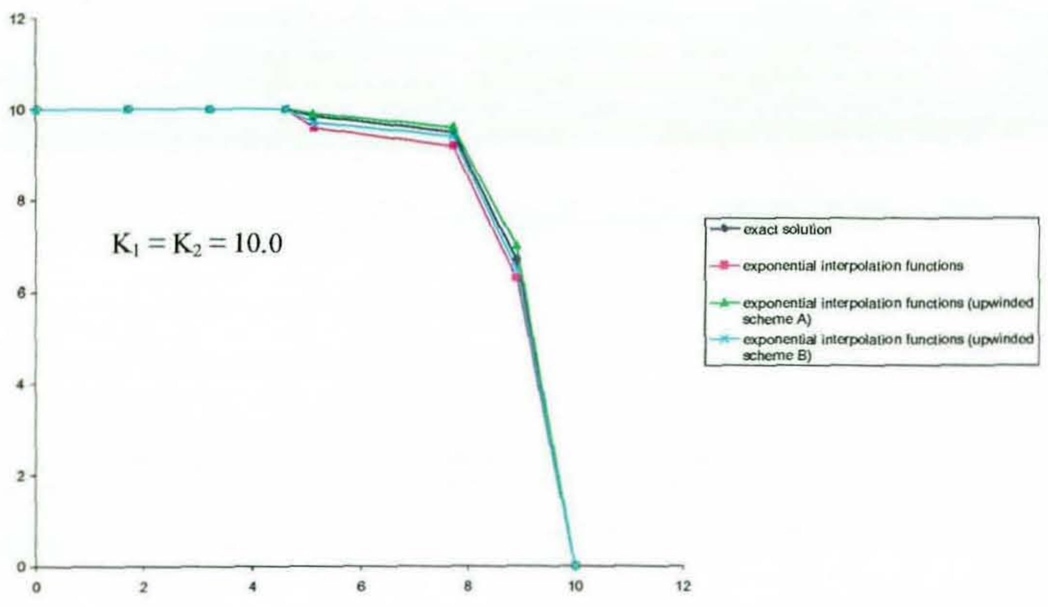


Figure 2.2 Solution along OP

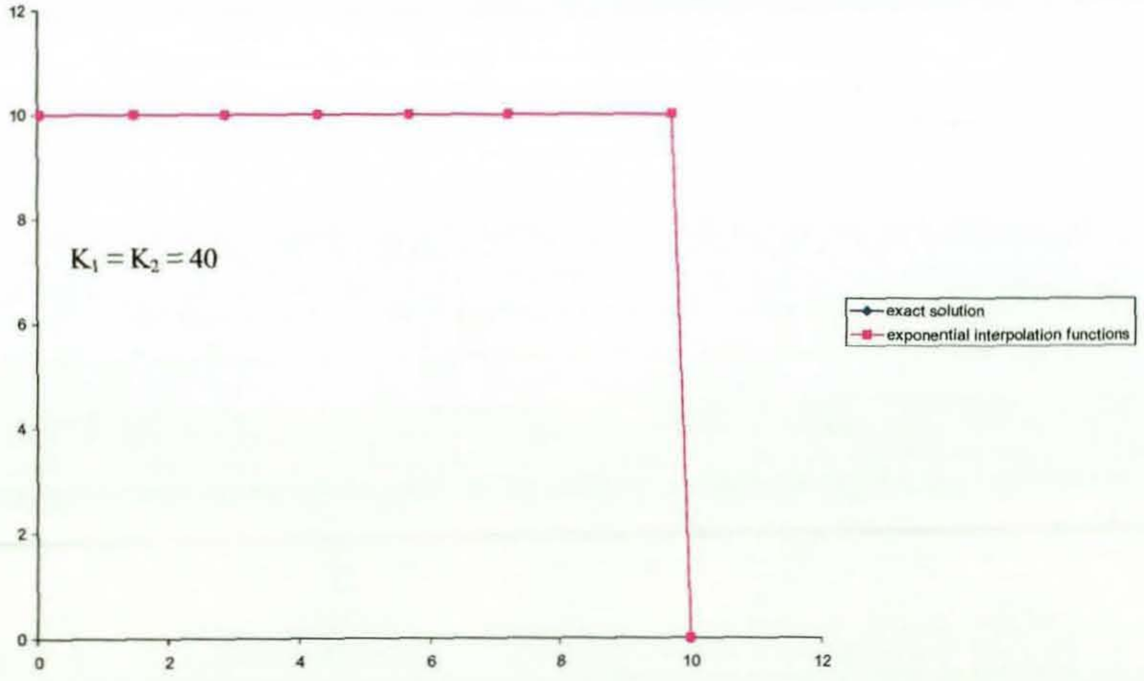


Figure 2.3 Solution along OP

2.4 Discussion

The solution for typical multi-physics model given by relatively complex of differential equations described in the previous sections of this chapter show that very accurate and smooth results can be generated for mathematical models expressed in terms of such equations by the use of exponential trial functions. At this point we emphasise that, although, the presented solutions, both using analytical and discretised finite element methods, are obtained for hyperbolic equations all of the discussions are valid for other types of differential models. It is indeed the case that the solution of elliptic and parabolic equations will be simpler and arguments supporting our conclusions will be more straightforward, (Zienkiewicz and Taylor, 1988).

Despite the proven rigor of the solutions obtained using exponential trial functions for all types of P.D.Es such techniques are not very practical. This is because that on one hand, we need discretisation techniques to incorporate complex boundary conditions in our solutions and resolve the issues related to irregular domain geometry and on the other hand, we need to prevent over constraining of the function compatibility at internal boundaries of a discretised domain. A simple analogy for this problem is the difficulty of fitting a curve through a large number of data points using a high order trial function. We note that the points in a data set can always be connected using straight lines without any parasitic wiggles but attempts to use high order curves will often fail due to over constraining of the compatibility of function at connecting points (Pittman, 1989). Mathematically this is referred to as the 'locking problem' (Burman and Hansbo, 2002).

The main objective of this research has been to explore the development of solution techniques for P.D.Es that resolve this problem and at the same time have the flexibility to be extended to general types of domains and boundary conditions. In the next chapter the technique used have been discussed in detail.

However, before starting such a discussion we need to consider the physical background of mathematical models that represent situations that locking may cause significant problems. We also need to explore in clearer physical terms the use of higher order trial functions within the development of practical solution schemes which, for example, do not rely on excessive mesh refinement.

2.5 Multi-scale Flow Problems: A Physical Description

Quantitative analysis of multi-scale problems has become an important issue in the engineering flow processes. Mathematical models of such flow regimes are oftem complex P.D.Es and their solution requires sophisticated numerical techniques. However, as the examples shown in this chapter indicate obtaining very accurate solutions for these equations is possible provided that certain complications have been resolved. The basic concept of a multi-scale problem is explained below via comparisons between the free and porous flow regimes with different physical properties.

Figure 2.4 shows a schematic diagram of a laminar plug flow where the domain is open to flow and its walls are not permeable. The flow is subject to perfect slip wall condition. In this case no stress is carried by the fluid. Such a free flow regime can be described mathematically by the use of Euler equations, (Aris, 1989).

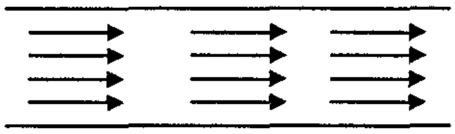


Figure 2.4 Plug flow regime (no slip wall)

In figure 2.5 the laminar free flow regime where the flow is subject to no slip wall conditions is shown. In this case the fluid carries all of the stress and becomes deformed. This flow regime can be described by Stokes or Navier-Stokes equation (depending on the Reynolds number).

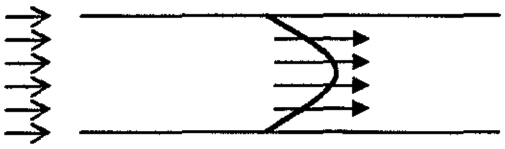


Figure 2.5 Velocity profile of Free flow regime (no permeability)

Figure 2.6 shows the physical features of a porous flow regime with high permeability (i.e. the domain consists of large pores). In this case the velocity at the walls is zero (i.e. no slip wall conditions). The fluid no longer carries all of the stress and some is borne by the porous medium. Such a porous flow phenomenon can be described mathematically by the Brinkman equation.

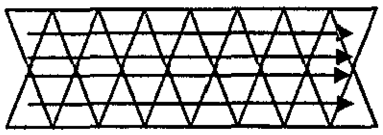


Figure 2.6 Porous flow regime (high permeability)

Figure 2.7 gives a representation of the physical aspects of a porous flow regime with very low permeability (i.e. the porous medium is dense and has very fine pores). In this case a slip wall condition is established and the velocity has a flat profile across the porous material. The stress is now carried completely by the solid matrix. Such a porous flow phenomenon can be described mathematically by the Darcy equation.

Note that although the velocity profile in this case will be similar to the one shown in figure 2.4 the mathematical representation of flow in the two cases will be very different. This is because the fluid viscosity plays no role in the free plug flow case and in contrast has a significant effect on the nature of a low permeability flow system.

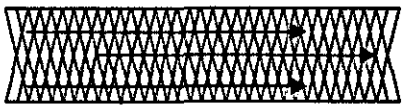


Figure 2.7 Porous flow regime (low permeability)

In figure 2.8 the typical velocity profile in a porous flow system where the permeability is high is shown. Amongst all of the regimes described here only the latter case can be regarded as a multi-scale flow problem. This is because the flow pattern at layers near the wall is very different in character to the established flow pattern within the domain. Inside the domain the profile will be plug flow but near the walls it will change very abruptly to a parabolic type.

Although Brinkman equation is able to characterize the flow in highly permeable porous medium with low Reynolds numbers the multi-scale nature of the flow makes it necessary to use excessive domain discretisation near the walls to obtain a good solution. Discussion presented in the previous section of this chapter is therefore directly relevant to this simulations of such flow systems.

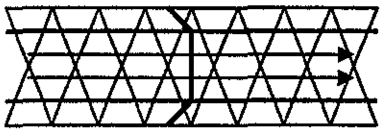


Figure 2.8 Velocity profile in a Porous flow regime with high permeability.

The standard Galerkin finite element method is not a strong enough approach for transport models displaying multi-scale behaviour (Donea and Heurta, 2003). For these problems, a particular class of sub-grid scale models are proposed which are known as multi-scale methods (Hughes, 1995; Hughes and Stewart 1996).

This is mainly due to the fact that the representation of all physical scales needs a high level of discretisation which is a common difficulty with these problems . If the discretisation at a coarse level ignores the fine scale then the solution will be unstable and inaccurate. The influence of the fine scales must be incorporated into the model. If the flow occurs in highly permeable porous media the thickness of the boundary layer decreases by reducing the permeability. The discretisation level must be less than the boundary layer thickness to achieve stable solution(Parvazinia et al., 2006). This problem can be satisfactorily resolved by the use of higher order approximation functions. Therefore if the problems related to 'numerical locking' can be resolved methods based on such trial functions will be the appropriate technique for multi-scale flow problems.

Similar multi-scale behaviour can be seen in turbulent flows, large scale molecular dynamic simulations, weather forecasting, reaction and convection dominated transport problems.

Chapter 3

Bubble Enriched Finite Element Discretisation of Multi-scale Field Problems

3.1 Introduction

As described in chapter 2 of this thesis multi-scale problems are common to many flow processes of engineering importance. In this chapter the use of bubble functions in conjunction with finite element discretisations of a variety of field problems are explained. As an engineering approach rather than a formal mathematical derivation is the aim of this research the development of bubble based finite element schemes is explained via examples.

Field problems in engineering are usually represented mathematically by sets of governing differential equations. In particular, for multidimensional or transient cases the governing model equations are derived as partial differential equations of elliptic, parabolic or hyperbolic type. Bubble enriched discretisations provide means of developing powerful and convenient finite element schemes for all three types of partial differential equations. As shown later in this chapter they are particularly useful in the solution of hyperbolic equations which represent convection dominated phenomena.

In the remainder of this chapter solutions for different types of equations based on bubble functions are given. Initially, in order to provide the most clear explanation of the technique one dimensional problems which can be solved almost without any heuristic extension of the method are discussed. This is followed by the extension of the methodology to two and three dimensional problems.

3.2 Finite element solution of a bench mark one dimensional problem

We start with the solution of the following problem. Consider the following differential equation:

$$\frac{d^2T}{dx^2} + a\frac{dT}{dx} + T = 0 \tag{3.1}$$

Over problem domain Ω_e (figure 3.1)x = [0 , 2]

subject to the boundary conditions given as

$$x = 0 \Rightarrow T=0$$

$$x = 2 \Rightarrow T = 1$$

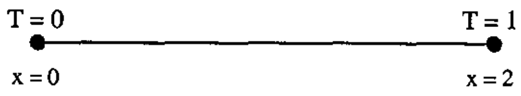


Figure 3.1 Problem Domain

where a = 50 (i.e. the first order derivative is significant).

Our objective is to obtain a solution for this problem using bubble function method and compare it with an ordinary finite element solution.

The analytical solution of this problem can be obtained using traditional methods (Jenson and Jeffreys,1983) and is written as:

$$T = 1.04008\left(e^{-0.02x} - e^{-49,98x}\right) \qquad (3.2)$$

3.2.1 Standard Galerkin Finite Element Method

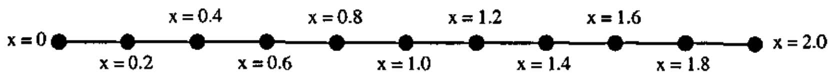


Figure 3.2 Discretised Domain

Let us first consider the Galerkin finite element solution of equation (3.1).

Following the discretisation of the solution domain Ω (figure 3.2) into 10 two-node Lagrange elements, and representation of T as $T = \sum N_i(x)t_i$ in terms of shape functions Ni(x), i=1,2 within a finite element space Ωe, the elemental Galerkin-weighted residual statement of the differential equation is written as

$$\int_{\Omega_e} N_j \left(\frac{d^2 \sum N_i(x)t_i}{dx^2} + a\frac{d\sum N_i(x)t_i}{dx} + \sum N_i(x)t_i \right) dx = 0 \qquad (3.3)$$

Where N_j is a weight function which in the Galerkin method is identical to N_i (i.e. shape functions). After the application of Green's theorem to the second order term in equation (3.3) we obtain the weak form of the above statement as

$$-\int_{\Omega_e}\left(\frac{d\sum N_i(x)t_i}{dx}.\frac{dN_j}{dx}\right)dx+a\int_{\Omega_e}\left(N_j\frac{d\sum N_i(x)t_i}{dx}\right)dx+\int_{\Omega_e}\left(N_j\sum N_i(x)t_i\right)dx+$$

$$N_j\frac{d\sum N_i(x)t_i}{dx}\bigg|_{\Gamma_e}=0 \qquad \textbf{(3.4)}$$

where Γ_e characterizes an element boundary. Based on equation (3.4) the elemental stiffness equation is formulated as

$$\begin{cases}-\int_{\Omega_e}\frac{d}{dx}(N_IT_I+N_{II}T_{II})\frac{dN_I}{dx}dx+a\int_{\Omega_e}\frac{d}{dx}(N_IT_I+N_{II}T_{II})N_Idx+\\ \int_{\Omega_e}(N_IT_I+N_{II}T_{II})N_Idx+N_I\phi|_{\Gamma_e}=0\\ -\int_{\Omega_e}\frac{d}{dx}(N_IT_I+N_{II}T_{II})\frac{dN_{II}}{dx}dx+a\int_{\Omega_e}\frac{d}{dx}(N_IT_I+N_{II}T_{II})N_{II}dx+\\ \int_{\Omega_e}(N_IT_I+N_{II}T_{II})N_{II}dx+N_{II}\phi|_{\Gamma_e}=0\end{cases} \qquad \textbf{(3.5)}$$

where ϕ represents the boundary line term (i.e. the flux through boundary). Using matrix notation equation (3.5) is written as

$$\begin{bmatrix}-\int_{\Omega_e}\left(\frac{dN_I}{dx}\frac{dN_I}{dx}-a\frac{dN_I}{dx}N_I-N_IN_I\right)dx & -\int_{\Omega_e}\left(\frac{dN_{II}}{dx}\frac{dN_I}{dx}-a\frac{dN_{II}}{dx}N_I-N_{II}N_I\right)dx\\ -\int_{\Omega_e}\left(\frac{dN_I}{dx}\frac{dN_{II}}{dx}-a\frac{dN_I}{dx}N_{II}-N_IN_{II}\right)dx & -\int_{\Omega_e}\left(\frac{dN_{II}}{dx}\frac{dN_{II}}{dx}-a\frac{dN_{II}}{dx}N_{II}-N_{II}N_{II}\right)dx\end{bmatrix}\begin{Bmatrix}T_I\\ T_{II}\end{Bmatrix}$$

$$=\begin{Bmatrix}-N_I\phi|_{\Gamma_e}\\ -N_{II}\phi|_{\Gamma_e}\end{Bmatrix} \qquad \textbf{(3.6)}$$

Note that the stiffness matrix obtained for this problem is not symmetric which is attributable to the existence of the first order derivative in the original equation. After the substitution for the shape functions and algebraic manipulations

$$\frac{-1}{l_e^2}\begin{pmatrix} \int_0^{l_e}\left[1+a(l_e-x)-(l_e-x)^2\right]dx & \int_0^{l_e}\left[-1-a(l_e-x)-x(l_e-x)\right]dx \\ \int_0^{l_e}-\left[1+ax-x(l_e-x)\right]dx & \int_0^{l_e}\left[1-ax-x^2\right]dx \end{pmatrix}\begin{pmatrix} T_I \\ T_{II} \end{pmatrix}=\begin{pmatrix} q_I \\ -q_{II} \end{pmatrix} \qquad \textbf{(3.7)}$$

After the evaluation of the integrals in the terms of the coefficient matrix, we have

$$\begin{pmatrix} \left(\frac{-1}{l_e}-\frac{a}{2}+\frac{l_e}{3}\right) & \left(\frac{1}{l_e}+\frac{a}{2}+\frac{l_e}{6}\right) \\ \left(\frac{1}{l_e}-\frac{a}{2}+\frac{l_e}{6}\right) & \left(\frac{-1}{l_e}+\frac{a}{2}+\frac{l_e}{3}\right) \end{pmatrix}\begin{pmatrix} T_I \\ T_{II} \end{pmatrix}=\begin{pmatrix} q_I \\ -q_{II} \end{pmatrix} \qquad \textbf{(3.8)}$$

Choosing a domain discretisation based on 10 elements of equal size ($l_e = 0.2$) we have

$$\begin{pmatrix} (-4.934-a/2) & (5.033+a/2) \\ ((5.033-a/2)) & (-4.934+a/2) \end{pmatrix}\begin{pmatrix} T_I \\ T_{II} \end{pmatrix}=\begin{pmatrix} q_I \\ -q_{II} \end{pmatrix} \qquad \textbf{(3.9)}$$

After t the assembly and insertion of the boundary conditions the following set of global stiffness equations is derived, for a = 50

$$\begin{bmatrix} d & c & & & & & & \bigcirc & \\ s & d & c & & & & & & \\ & s & d & c & & & & & \\ & & s & d & c & & & & \\ & & & s & d & c & & & \\ & \bigcirc & & & s & d & c & & \\ & & & & & s & d & c & \\ & & & & & & s & d & c \\ & & & & & & & s & d \end{bmatrix}\begin{bmatrix} T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \\ T_{10} \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -c \end{bmatrix} \qquad \textbf{(3.10)}$$

where d = -9.868,c = 30.033 and s = 19.967. We can now obtain the solution for this problem and compare it with the analytical result.

The first order derivative in equation (3.1) corresponds to the convection in a field problem, where a gives an indication of the rate by which convection is taking place within the domain. The examples shown in the figure 3.3 comparing the analytical solution to the Galerkin solutions of the field problem with varying amounts of convection taking place (i.e. the value of a increasing), show the inability of the standard Galerkin method to produce meaningful results for convection-dominated equations.

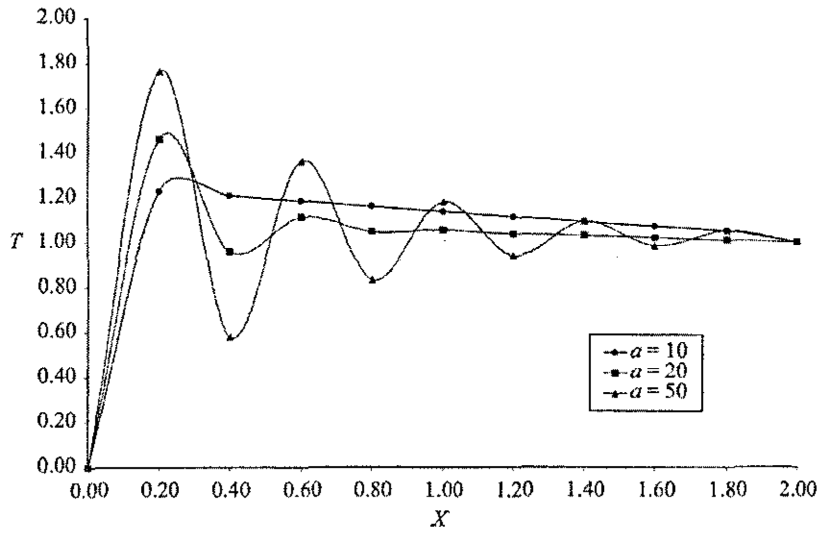


Figure 3.3 Galerkin solution with varying values of convection

3.2.2 Petrov-Galerkin (Upwinding) Method

To resolve the difficulty of the solution of hyperbolic (convection-dominated) equations, upwinding or Petrov-Galerkin methods are employed. To demonstrate the application of upwinding we consider the case where only the weight function applied to the first order derivative in the weak variational statement of the problem, represented by equation (3.3), is modified.

The weighted residual statement corresponds to equation (3.1) is hence written as

$$\int_{\Omega_e} N_j\left(\frac{d^2\sum N_i(x)t_i}{dx^2}+\sum N_i(x)t_i\right)dx + N_j^*\left(a\frac{d\sum N_i(x)t_i}{dx}\right)dx = 0 \qquad \textbf{(3.11)}$$

Integration by part (i.e. application of Green's theorem to the second order term in equation (3.11) gives the weak form of the problem as

$$\begin{aligned}&-\int_{\Omega_e}\left(\frac{d\sum N_i(x)t_i}{dx}.\frac{dN_j}{dx}\right)dx+\int_{\Omega_e}\left(N_j\sum N_i(x)t_i\right)dx+N_j\frac{d\sum N_i(x)t_i}{dx}\bigg|_{\Gamma}\\&+a\int_{\Omega_e}\left(N_j^*\frac{d\sum N_i(x)t_i}{dx}\right)dx=0\end{aligned} \qquad \textbf{(3.12)}$$

Using two-noded Lagrangian elements the shape functions are given as

$$N_I=\frac{l_e-x}{l_e} \quad \text{and} \quad N_{II}=\frac{x}{l_e}$$

therefore

$$\frac{dN_I}{dx}=-\frac{1}{l_e} \quad \text{and} \quad \frac{dN_{II}}{dx}=\frac{1}{l_e}$$

In the simple one dimensional example considered here the upwinded weight function found using the analytical solution (3.2) of equation (3.1) with a= 1,

$$T=2.754e^{-\frac{1}{2}x}\sin\frac{\sqrt{3}}{2}x \qquad \textbf{(3.13)}$$

is reduced to $W=N+\beta(dN/dx)$. Therefore, the modified weight functions applied to the first order derivative term in equation (3.12) can be written as

$$\begin{cases} N_I^* = N_I + \beta \dfrac{dN_I}{dx} = \dfrac{l_e - x}{l_e} - \dfrac{\beta}{l_e} = \dfrac{l_e - x - \beta}{l_e} \\ N_{II}^* = N_{II} + \beta \dfrac{dN_{II}}{dx} = \dfrac{x}{l_e} + \dfrac{\beta}{l_e} = \dfrac{x + \beta}{l_e} \end{cases} \tag{3.14}$$

The general elemental stiffness equation can thus be written as

$$\begin{bmatrix} -\int\limits_{\Omega_e}\left(\dfrac{dN_I}{dx}\dfrac{dN_I}{dx} - a\dfrac{dN_I}{dx}N_I^* - N_I N_I\right)dx & -\int\limits_{\Omega_e}\left(\dfrac{dN_{II}}{dx}\dfrac{dN_I}{dx} - a\dfrac{dN_{II}}{dx}N_I^* - N_{II} N_I\right)dx \\ -\int\limits_{\Omega_e}\left(\dfrac{dN_I}{dx}\dfrac{dN_{II}}{dx} - a\dfrac{dN_I}{dx}N_{II}^* - N_I N_{II}\right)dx & -\int\limits_{\Omega_e}\left(\dfrac{dN_{II}}{dx}\dfrac{dN_{II}}{dx} - a\dfrac{dN_{II}}{dx}N_{II}^* - N_{II} N_{II}\right)dx \end{bmatrix} \begin{Bmatrix} T_I \\ T_{II} \end{Bmatrix}$$

$$= \begin{Bmatrix} -N_I \phi\big|_{\Gamma_e} \\ -N_{II} \phi\big|_{\Gamma_e} \end{Bmatrix} \tag{3.15}$$

Substitution of the shape functions gives

$$\frac{-1}{l_e^2}\begin{pmatrix} \int\limits_0^{l_e}\left[1 + a(l_e - x - \beta) - (l_e - x)^2\right]dx & \int\limits_0^{l_e}\left[-1 - a(l_e - x - \beta) - x(l_e - x)\right]dx \\ \int\limits_0^{l_e} -\left[1 + a(x + \beta) - x(l_e - x)\right]dx & \int\limits_0^{l_e}\left[1 - a(x + \beta) - x^2\right]dx \end{pmatrix}\begin{pmatrix} T_I \\ T_{II} \end{pmatrix} = \begin{pmatrix} q_I \\ -q_{II} \end{pmatrix} \tag{3.16}$$

After integration

$$\begin{pmatrix} \left(\dfrac{-1}{l_e} + \dfrac{a\beta}{l_e} - \dfrac{a}{2} + \dfrac{l_e}{3}\right) & \left(\dfrac{1}{l_e} - \dfrac{a\beta}{l_e} + \dfrac{a}{2} + \dfrac{l_e}{6}\right) \\ \left(\dfrac{1}{l_e} - \dfrac{a}{2} - \dfrac{a\beta}{l_e} + \dfrac{l_e}{6}\right) & \left(\dfrac{-1}{l_e} + \dfrac{a\beta}{l_e} + \dfrac{a}{2} + \dfrac{l_e}{3}\right) \end{pmatrix}\begin{pmatrix} T_I \\ T_{II} \end{pmatrix} = \begin{pmatrix} q_I \\ -q_{II} \end{pmatrix} \tag{3.17}$$

Choosing a mesh of 10 elements of equal size we have

$$\begin{bmatrix} -4.933 + 5.0a\beta - a/2 & 5.033 + a/2 - 5.0a\beta \\ 5.033 - a/2 - 5.0a\beta & -4.933 + 5.0a\beta + a/2 \end{bmatrix}\begin{pmatrix} T_I \\ T_{II} \end{pmatrix} = \begin{pmatrix} q_I \\ -q_{II} \end{pmatrix} \tag{3.18}$$

We consider the solution of equation (3.1) with value of a = 50, in this case the general form of the elemental stiffness equation is written as

$$\begin{bmatrix} -29.933+250.0\beta & 30.033-250.0\beta \\ -19.967-250.0\beta & 20.067+250.0\beta \end{bmatrix}\begin{pmatrix} T_I \\ T_{II} \end{pmatrix}=\begin{pmatrix} q_I \\ -q_{II} \end{pmatrix} \tag{3.19}$$

After the assembly of the elemental equations into a global set and imposition of the boundary conditions the final solution of the original differential equation with respect to various values of upwinding parameter β can be found. The analytical solution of equation (3.1) with a=50 is found as equation (3.2). The finite element results obtained for various values of β are compared with the analytical solution in figure 3.4. As can be seen using a value of β = 0.5 a stable numerical solution is obtained. However, this solution is over-damped and inaccurate. Therefore the main problem is to find a value of upwinding parameter that eliminates oscillations without generating over-damped results.

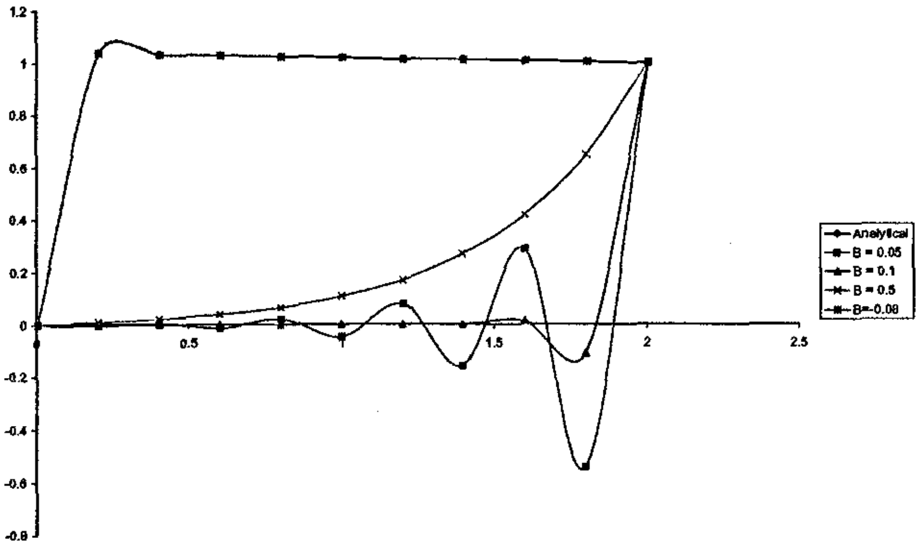


Figure 3.4 Upwinding Solutions in comparison with Analytical Solution

The numerical experiments show that a value of $\beta = -0.08$ gives a solution to equation (3.1) which is exactly comparable with the analytical results (i.e. super convergent). However, extension of this method to multi-dimensional problems is not straight forward and involves arbitrary approximations (Nassehi, 2002).

3.2.3 Bubble Function Method

We repeat the solution of equation (3.1), again using the Galerkin finite element technique. However, this time we consider a one dimensional element as shown in figure 3.5. Although there are three nodes in this element we take the interpolation functions associated with nodes 1 and 2 (i.e. corresponding to degrees of freedom t1 and t2) as ordinary linear Lagrangian shape functions. We define a special shape function corresponding to the middle node (i.e. relevant to the degree of freedom t3) as $N_B = \left(1 - \xi^{10}\right)$ using a local elemental coordinate system ξ .

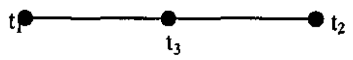


Figure 3.5 Bubble enriched finite element

Therefore approximation of the unknown field variable in equation (3.1) is represented as

$$T \approx \tau = \sum_{i=1}^{3} N_i(\xi) t_i \tag{3.20}$$

or

$$\tau = \tfrac{1}{2}(1-\xi)t_1 + \tfrac{1}{2}(1+\xi)t_2 + \left(1-\xi^{10}\right)t_3 \tag{3.21}$$

[where ξ is taken as -1 (at t_1) and $+1$ (at t_2) and in the bubble part as equal to 0 (at t_3)].

The above approximation can be compared with an ordinary linear discretisation simply written as

$$\bar{\tau} = \tfrac{1}{2}(1-\xi)t_1 + \tfrac{1}{2}(1+\xi)t_2 \qquad (3.22)$$

Which is used in the standard Galerkin method described previously. The important point to note is that the introduction of the additional degree of freedom does not affect the basic structure of the solution procedure or the degree of the inter-element continuity as it will disappear at nodes 1 and 2 (i.e. exterior boundaries of the element). Therefore, this function (called the 'bubble function')

$$N_B = N_3 = \left(1-\xi^{10}\right)t_3 \qquad (3.23)$$

significantly increases the accuracy of the approximation of the field unknown over an element domain without altering the basic properties of the numerical scheme. At this point it appears as we have chosen this function arbitrarily. Later in this chapter systematic methods for obtaining appropriate bubble functions are discussed in detail. However here we continue with the development of a Galerkin finite element solution for the present benchmark problem using the discretisation described be equation (3.21) to illustrate the solution procedure.

3.2.4 Galerkin Finite Element solution using Bubble Function approach

We follow the normal procedure of the standard Galerkin method to formulate a residual statement for equation (3.1). As the use of bubble enriched elements has not changed the inter-element continuity properties of the discretised domain we need to apply integration by parts to the formulated residual statement this gives

$$-\int \frac{d\tau}{dx}\frac{dw}{dx} + a\int \frac{d\tau}{dx} w dx + \int \tau w dx + \phi w| = 0 \qquad (3.24)$$

We now use the decomposition procedure to extract the bubble part and reassemble the element approximation as

$$\tau = N_1 t_1 + N_2 t_2 + N_3 t_3 = N_4 t_1 + N_5 t_2 \qquad (3.25)$$

This process can be explained as follows. We consider the integration of equation (3.24) over the space of an element

$$\int_0^L (a\tau_x w - \tau_x w_x + \tau w)dx = 0 \qquad (3.26)$$

Or using a local elemental co-ordinate system (ξ) we have

$$\int_{-1}^{1} (-\tfrac{2}{h}\tau_\xi \tfrac{2}{h} w_\xi + a\tfrac{2}{h}\tau_\xi w + \tau w)\tfrac{h}{2} d\xi = 0 \qquad (3.27)$$

Substitute from the following approximations

$$\tau = \tfrac{1}{2}(1-\xi)t_1 + \tfrac{1}{2}(1+\xi)t_2 + \tfrac{1}{2}(1-\xi^{10})t_3 \qquad (3.28)$$

and

$$\frac{d\tau}{dx} = \frac{2}{h}\left(\frac{d\tau}{d\xi}\right) = \frac{2}{h}\left(\frac{t_2}{2} - \frac{t_1}{2} - 10\xi^9\right) \qquad (3.29)$$

and

$$dx = \tfrac{h}{2} d\xi \qquad (3.30)$$

Using a weight function which is identical to the bubble function as

$$w_B = \left(1-\xi^{10}\right) \tag{3.31}$$

we have

$$\frac{dw_B}{dx} = \frac{2}{h}\left(\frac{dw_B}{d\xi}\right) = \frac{2}{h}\left(-10\xi^9\right) \tag{3.32}$$

After substituting equations (3.28) to (3.32) into equation (3.26) the following integrals over an element domain are obtained

$$\int_0^L aw\tau_x dx = a\int_{-1}^{+1}\left(1-\xi^{10}\right)\left(\frac{t_2}{2}-\frac{t_1}{2}-10\xi^9\right)d\xi = \tfrac{20}{22}at_2 - \tfrac{20}{22}at_1 \tag{3.33}$$

and

$$\int_0^L w_x\tau_x dx = \int_{-1}^{+1}\left(-10\xi^9\right)\left(\frac{t_2}{2}-\frac{t_1}{2}-10\xi^9\right)d\xi = -\tfrac{400}{19h}t_3 \tag{3.34}$$

and

$$\int_0^L w\tau dx = \int_{-1}^{+1}\left(1-\xi^{10}\right)\left(\tfrac{1}{2}(1-\xi)t_1 + \tfrac{1}{2}(1+\xi)t_2 + \tfrac{1}{2}\left(1-\xi^{10}\right)t_3\right)d\xi = \tfrac{5h}{11}t_1 + \tfrac{5h}{11}t_2 + \tfrac{200h}{231}t_3 \tag{3.35}$$

and

$$\int_0^L aw\tau_x - w_x\tau_x + w\tau dx = -\tfrac{10}{11}at_1 + \tfrac{10}{11}at_2 - \tfrac{400}{19h}t_3 + \tfrac{5h}{11}t_1 + \tfrac{5h}{11}t_2 + \tfrac{200h}{231}t_3 \tag{3.36}$$

Using the set of simultaneous equations (3.33) to (3.36) gives the following expression for t_3 in terms of t_1 and t_2.

$$t_3 = \frac{h-2a}{\frac{880}{19h}-\frac{40h}{21}}t_1 + \frac{h+2a}{\frac{880}{19h}-\frac{40h}{21}}t_2 \tag{3.37}$$

$$\text{Let } b_1 = \frac{h-2a}{\frac{880}{19h}-\frac{40h}{21}} \text{ and } -b_2 = -\frac{h+2a}{\frac{880}{19h}-\frac{40h}{21}} \tag{3.38}$$

After condensing the variable τ in equation (3.21) via the substitution of the interpolation functions we obtain

$$\tau = \left(\tfrac{1}{2}(1-\xi)+\left(1-\xi^{10}\right)b_1\right)t_1 + \left(\tfrac{1}{2}(1+\xi)-\left(1-\xi^{10}\right)b_2\right)t_2 \tag{3.39}$$

Comparison of equations (3.39) and (3.25) gives

$$N_4 = \left(\tfrac{1}{2}(1-\xi)+\left(1-\xi^{10}\right)b_1\right) \tag{3.40}$$

$$N_5 = \left(\tfrac{1}{2}(1+\xi)-\left(1-\xi^{10}\right)b_2\right) \tag{3.41}$$

Expression of the interpolation model in terms of two shape functions, as given in equation (3.25), allows the use of the Galerkin method in the usual manner.

Differentiating the above interpolation functions given in equations (3.40) and (3.41) we have

$$\frac{dN_4}{dx} = -\frac{1}{2} - 10\xi^9 b_1 \text{ and } \frac{dN_5}{dx} = \frac{1}{2} + 10\xi^9 b_2 \tag{3.42}$$

Taking the weight functions to be the same as the interpolation functions

$$w_1 = \tfrac{1}{2}(1-\xi) \tag{3.43}$$

$$w_2 = \tfrac{1}{2}(1+\xi) \tag{3.44}$$

Differentiating the weight functions

$$w_1 = -\tfrac{1}{2} \tag{3.45}$$

$$w_2 = \tfrac{1}{2} \tag{3.46}$$

Using the modified functions we now have

$$\frac{dT^2}{dx^2} + a\frac{dT}{dx} + T \Rightarrow \int \left(a\frac{d\tau}{dx}w - \frac{d\tau}{dx}\frac{dw}{dx} + \tau w \right)dx = 0 \tag{3.47}$$

and

$$\begin{gathered}\int_0^L \left(a\frac{dN_4}{dx}w_1 - \frac{dN_4}{dx}\frac{dw_1}{dx} + N_4 w_1 \right)dx = 0 \\ \Rightarrow \int_{-1}^{+1} \left(\left(-a\left(\tfrac{1}{2} + 10\xi^9 b_1\right)\tfrac{1}{2}(1-\xi)\right) - \left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2} + 10\xi^9 b_1\right)\right) + \left(\tfrac{h}{4}\left((1-\xi) + \left(1-\xi^{10}\right)b_1\right)(1-\xi)\right)\right)d\xi\end{gathered} \tag{3.48}$$

Therefore

$$\begin{gathered}\int_0^L \left(a\frac{dN_5}{dx}w_1 - \frac{dN_5}{dx}\frac{dw_1}{dx} + N_5 w_1 \right)dx = 0 \\ \Rightarrow \int_{-1}^{+1} \left(\left(a\left(\tfrac{1}{2} + 10\xi^9 b_2\right)\tfrac{1}{2}(1-\xi)\right) + \left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2} + 10\xi^9 b_2\right)\right) + \left(\tfrac{h}{4}\left((1+\xi) - \left(1-\xi^{10}\right)b_2\right)(1-\xi)\right)\right)d\xi\end{gathered} \tag{3.49}$$

and

$$\begin{gathered}\int_0^L \left(a\frac{dN_4}{dx}w_2 - \frac{dN_4}{dx}\frac{dw_2}{dx} + N_4 w_2 \right)dx = 0 \\ \Rightarrow \int_{-1}^{+1} \left(\left(-a\left(\tfrac{1}{2} + 10\xi^9 b_1\right)\tfrac{1}{2}(1+\xi)\right) + \left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2} + 10\xi^9 b_1\right)\right) + \left(\tfrac{h}{4}\left((1-\xi) + \left(1-\xi^{10}\right)b_1\right)(1+\xi)\right)\right)d\xi\end{gathered} \tag{3.50}$$

also

$$\int_0^L \left(a\frac{dN_5}{dx} w_2 - \frac{dN_5}{dx}\frac{dw_2}{dx} + N_5 w_2 \right) dx = 0$$

$$\Rightarrow \int_{-1}^{+1} \left(\left(a\left(\tfrac{1}{2} + 10\xi^9 b_2\right)\tfrac{1}{2}(1+\xi)\right) - \left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2} + 10\xi^9 b_2\right)\right) + \left(\tfrac{h}{4}\left((1+\xi) - \left(1-\xi^{10}\right)b_2\right)(1+\xi)\right)\right) d\xi \tag{3.51}$$

The above derived elemental integrals should now be evaluated. However in order to obtain the integrals we need to use the Gaussian quadrature. In contrast to normal finite element discretisation here we cannot use 2 or 3 point quadratures. It is necessary to use quadrature of higher degrees. In what follows, this procedure is explained.

3.3 Higher Order Gaussian Quadrature

Gaussian quadrature is used for the numerical integration to solve the integrals (3.48), (3.49), (3.50) and (3.51). The idea consists of a formula of n terms (dependent on the degree of the polynomial requiring integration) with n parameters, coefficients (weighting factors) applying these to each of the functional values. For example a formula of 3 terms will contain 6 parameters (three x terms and three weighting functions) this will correspond to an interpolating polynomial of degree 5 (Gerald and Wheatly,1999).

The Degree of the polynomials requiring integration goes up to a degree of 12 so the number of Gaussian points used is 7. the Validity of using 7 Gaussian points is up to degree 13.

The Legendre Polynomials from which the values of the roots are acquired.

$$(n+1)L_{n+1}(x) - (2n+1)xL_n(x) + nL_{n-1}(x) = 0 \tag{3.52}$$

with $L_0(x) = 1$ and $L_1(x) = x$ giving

$$L_2(x) \Rightarrow \frac{3xL_1(x) - (1)L_0(x)}{2} = \frac{3}{2}x^2 - \frac{1}{2} = 0 \tag{3.53}$$

$$L_3(x) \Rightarrow \frac{5xL_2(x)+(2)L_1(x)}{3} = \frac{5x^3-3x}{2} = 0 \qquad (3.54)$$

$$L_4(x) \Rightarrow \frac{7xL_3(x)+(3)L_2(x)}{4} = \frac{35x^4-30x^2+3}{8} = 0 \qquad (3.55)$$

$$L_5(x) \Rightarrow \frac{9xL_5(x)+(4)L_3(x)}{5} = \frac{63x^5-70x^3+15x}{8} = 0 \qquad (3.56)$$

$$L_6(x) \Rightarrow \frac{11xL_5(x)+(4)L_3(x)}{6} = \frac{231x^6-315x^4+105x^2-5}{16} = 0 \qquad (3.57)$$

$$L_7(x) \Rightarrow \frac{13xL_6(x)+(5)L_5(x)}{7} = \frac{429x^7-693x^5+315x^3-35x}{16} = 0 \qquad (3.58)$$

Then working out the roots of these Polynomials gives

For $L_2(x)$ the roots are ± 0.57735

For $L_3(x)$ the roots are $0, \pm 0.77460$

For $L_4(x)$ the roots are $\pm 0.33998, \pm 0.86114$

For $L_5(x)$ the roots are $0, \pm 0.53847, \pm 0.90618$

For $L_6(x)$ the roots are $\pm 0.23862, \pm 0.66121, \pm 0.93247$

For $L_7(x)$ the roots are $0, \pm 0.40585, \pm 0.74153, \pm 0.94911$

The weights are solved by the following set of simultaneous equations derived from the equation below

$$\int_{-1}^{+1} f(r) = w_1 f(r_1) + w_2 f(r_2) + w_3 f(r_3) + w_4 f(r_4) + w_5 f(r_5) + w_6 f(r_6) + w_7 f(r_7) \qquad (3.59)$$

where ri are the different roots of $L_7(x)$ and wi are the weights.

$$\int_{-1}^{+1} r^7 dr = 0 = w_1(r_1^7) + w_2(r_2^7) + w_3(r_3^7) + w_4(r_4^7) + w_5(r_5^7) + w_6(r_6^7) + w_7(r_7^7) \quad (3.60)$$

$$\int_{-1}^{+1} r^6 dr = \tfrac{2}{7} = w_1(r_1^6) + w_2(r_2^6) + w_3(r_3^6) + w_4(r_4^6) + w_5(r_5^6) + w_6(r_6^6) + w_7(r_7^6) \quad (3.61)$$

$$\int_{-1}^{+1} r^5 dr = 0 = w_1(r_1^5) + w_2(r_2^5) + w_3(r_3^5) + w_4(r_4^5) + w_5(r_5^5) + w_6(r_6^5) + w_7(r_7^5) \quad (3.62)$$

$$\int_{-1}^{+1} r^4 dr = \tfrac{2}{5} = w_1(r_1^4) + w_2(r_2^4) + w_3(r_3^4) + w_4(r_4^4) + w_5(r_5^4) + w_6(r_6^4) + w_7(r_7^4) \quad (3.63)$$

$$\int_{-1}^{+1} r^3 dr = 0 = w_1(r_1^3) + w_2(r_2^3) + w_3(r_3^3) + w_4(r_4^3) + w_5(r_5^3) + w_6(r_6^3) + w_7(r_7^3) \quad (3.64)$$

$$\int_{-1}^{+1} r^2 dr = \tfrac{2}{3} = w_1(r_1^2) + w_2(r_2^2) + w_3(r_3^2) + w_4(r_4^2) + w_5(r_5^2) + w_6(r_6^2) + w_7(r_7^2) \quad (3.65)$$

$$\int_{-1}^{+1} r dr = 0 = w_1 r_1 + w_2 r_2 + w_3 r_3 + w_4 r_4 + w_5 r_5 + w_6 r_6 + w_7 r_7 \quad (3.66)$$

$$\int_{-1}^{+1} dr = 2 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \quad (3.67)$$

The weights of $L_7(x)$ are

0	0.41796
±0.40585	0.38183
±0.74153	0.27971
±0.94911	0.12949

Evaluating the integrals by Gaussian Quadrature the following values are reached:

$$\int_{-1}^{+1}\left(\left(-a\left(\tfrac{1}{2}+10\xi^9 b_1\right)\tfrac{1}{2}(1-\xi)\right)-\left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2}+10\xi^9 b_1\right)\right)+\left(\tfrac{h}{4}\left((1-\xi)+\left(1-\xi^{10}\right)b_1\right)(1-\xi)\right)\right)d\xi \tag{3.68}$$

$$= -49.59369$$

$$\int_{-1}^{+1}\left(\left(a\left(\tfrac{1}{2}+10\xi^9 b_2\right)\tfrac{1}{2}(1-\xi)\right)+\left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2}+10^9 b_2\right)\right)+\left(\tfrac{h}{4}\left((1+\xi)-\left(1-\xi^{10}\right)b_2\right)(1-\xi)\right)\right)d\xi \tag{3.69}$$

$$= 49.77249$$

$$\int_{-1}^{+1}\left(\left(-a\left(\tfrac{1}{2}+10\xi^9 b_1\right)\tfrac{1}{2}(1+\xi)\right)+\left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2}+10\xi^9 b_1\right)\right)+\left(\tfrac{h}{4}\left((1-\xi)+\left(1-\xi^{10}\right)b_1\right)(1+\xi)\right)\right)d\xi \tag{3.70}$$

$$= -0.38479$$

$$\int_{-1}^{+1}\left(\left(a\left(\tfrac{1}{2}+10\xi^9 b_2\right)\tfrac{1}{2}(1+\xi)\right)-\left(\left(\tfrac{1}{h}\right)\left(\tfrac{1}{2}+10\xi^9 b_2\right)\right)+\left(\tfrac{h}{4}\left((1+\xi)-\left(1-\xi^{10}\right)b_2\right)(1+\xi)\right)\right)d\xi \tag{3.71}$$

$$= 0.40631$$

The evaluation of the integrals was necessary to be done by Gaussian Quadrature due to the fact that the solution of 2 dimensional and 3 dimensional problems would require Gaussian Quadrature which allows the evaluation of elemental stiffness matrix by a computer.

The following comparisons prove that if a quadrature of sufficient order is used results which are comparable with the analytical solution can be obtained (figure 3.6). In figure 3.7 we compare the described results obtained using bubble function with the standard Galerkin and Petrov-Galerkin solutions of equation (3.1). As shown in the figure 3.7 bubble enriched discretisation generates smooth non-oscillatory and accurate results for convection dominated case of a = 50 without the use of upwinding (i.e. Petrov-Galerkin discretisation). The advantage of the bubble function method over upwinding is in its flexibility for the extension into multi-dimensional problems. It is well known that the extension of streamline upwinding Petrov-Galerkin to multi-dimensional cases involves arbitrary approximations.

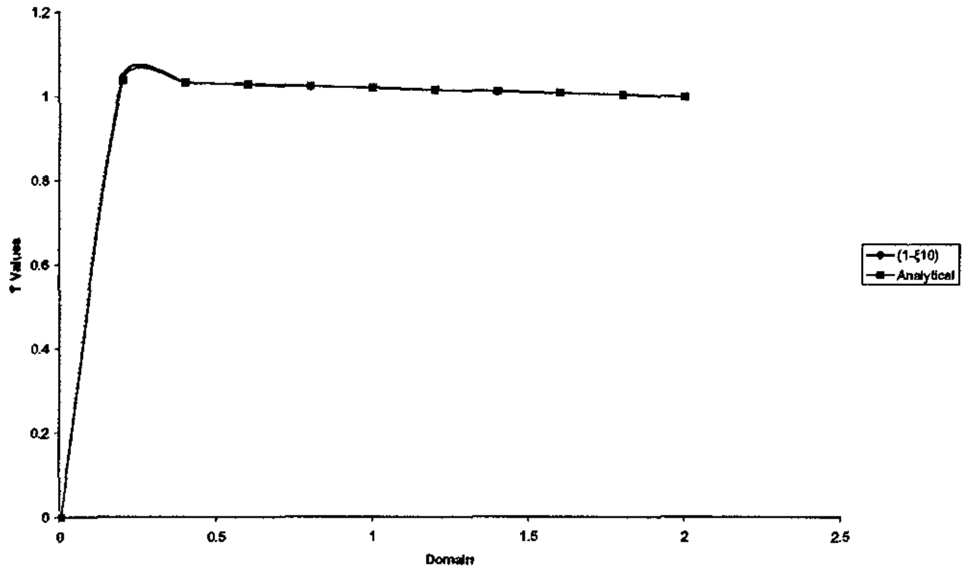


Figure 3.6 Bubble function solution in comparison with Analytical solution

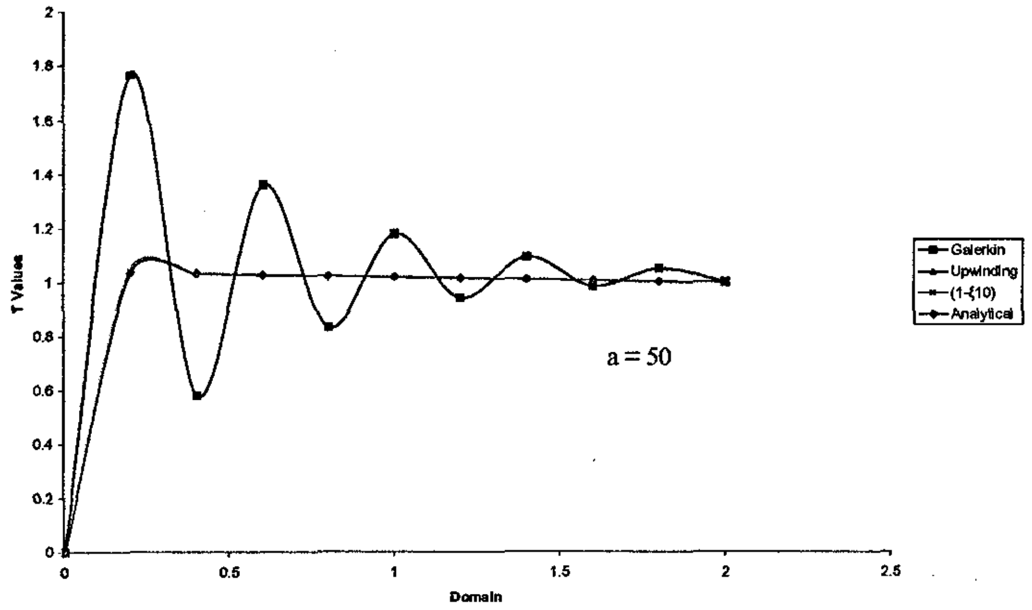


Figure 3.7 Comparison of the solution of the field problem

3.4 Extension of Bubble Function approach to multi dimensional problems

When extending the bubble function method to solve multi-scale problems, the unknown variable, which in one dimension was divided into the Galerkin part and the bubble part are considered to represent different scales in a multi-scale problem. Therefore the basic approximation of a field unknown over the space of an element is given as

$$u_h = u_1 + u_b \qquad \textbf{(3.72)}$$

where u_b is known as the fine scale (the bubble part) and u_1 is known as coarse scale (the standard Galerkin part) representing a standard finite element approximation polynomial (interpolation function). In the multi-scale variational formulation which is developed by Hughes, for the sub-grid model we consider a Dirichlet problem as:

$$\begin{cases} Lu_h = f & \text{in } \Omega \text{ (domain)} \\ u_h = g & \text{on } \Gamma \text{ (domain boundary)} \end{cases}$$

where L is a differential operator. Using the definition of a bilinear form as $a(.,.)$ the variational formulation for the above equation is

$$a(v_h, u_h) = (v_h, Lu_h)$$

where (. , .) represents a scalar product.

Let

$$u_h = u_1 + u_b$$
$$v_h = v_1 + v_b$$

where we assume

$$u_b = v_b = 0 \quad \text{on } \Gamma_e \text{ (sub}-\text{domain boundary)}.$$

The variational formulation may be written as

$$a(v_h,u_h)=(v_h,f) \text{ or } a(v_1+v_b,u_1+u_b)=(v_1+v_b,f)$$

it can be written as two sub-problems

$$a(v_b,u_1)+a(v_b,u_b)=(v_b,f) \tag{3.73}$$

$$a(v_1,u_1)+a(v_1,u_b)=(v_1,f) \tag{3.74}$$

The Euler-Lagrange equations of the first sub-problem is

$$\begin{cases} Lu_b=-(Lu_1-f) & \text{in } \Omega_e \text{ (sub-domain)} \\ u_b=0 & \text{on } \Gamma_e \text{ (sub-domain boundary)} \end{cases} \tag{3.75}$$

The extension of the bubble function approach to multidimensional problems uses the Green's function approach to solve the above equation.

3.5 Residual Free Bubble Function Method

This method offers a systematic approach for the extension of the bubble function enriched finite elements to two and three dimensional problems. However it can also be viewed as a mathematical method for the derivation of one dimensional bubble enriched elements. To solve equation (3.75), based on the residual free bubble function method the fine scale is divided into two parts as:

$$u_b=u_b^0+u_b^f \tag{3.76}$$

u_b^0 and u_b^f are, respectively, solutions of the following equations:

$$\begin{cases} Lu_b^0=-Lu_1 & \text{in } \Omega_e \\ u_b^0=0 & \text{on } \Gamma_e \end{cases} \tag{3.77}$$

$$\begin{cases} Lu_b^f = f & \text{in } \Omega_e \\ u_b^f = 0 & \text{on } \Gamma_e \end{cases} \tag{3.78}$$

Assuming that ϕ is a bubble shape function and ψ is a polynomial shape function, then equations (3.77) and (3.78) can be rewritten as:

$$\begin{cases} L\phi_i = -L\psi_i & \text{in } \Omega_e \\ \phi_i = 0 & \text{on } \Gamma_e \end{cases} \tag{3.79}$$

where ψ_i and ϕ_i are functions associated with node i. Ω_e is the element domain and Γ_e is the element boundary. Hence

$$\begin{cases} L\phi_f = f & \text{in } \Omega_e \\ \phi_f = 0 & \text{on } \Gamma_e \end{cases} \tag{3.80}$$

And

$$u_h = u_1 + u_b = \sum_{i=1}^{n} u_i(\psi_i + \phi_i) + \phi_f \tag{3.81}$$

where n is the number of nodes per element. To solve equations (3.80) and (3.81) it is assumed that:

$$N_i = \psi_i + \phi_i \tag{3.82}$$

Substituting equation (3.82) into equation (3.79), for a linear element on each node we have

$$\begin{cases} \dfrac{d^2 N_1}{dx^2} - \dfrac{1}{D_a} N_1 = 0 \quad \text{for } x \in [0 - l] \\ N_1 = \psi_1 \Rightarrow \begin{cases} N_1(0) = 1 \\ N_1(l) = 0 \end{cases} \end{cases} \tag{3.83}$$

$$\begin{cases} \dfrac{d^2N_2}{dx^2} - \dfrac{1}{D_a}N_2 = 0 \quad \text{for } x \in [0-l] \\ N_2 = \psi_2 \Rightarrow \begin{cases} N_2(0) = 0 \\ N_2(l) = 1 \end{cases} \end{cases} \tag{3.84}$$

Given that l is the elemental length and ψ_i is a linear shape function. The equation above is evaluated, giving bubble shape functions expressed in a local elemental coordinate system as:

$$\begin{cases} N_1 = \dfrac{\sinh\sqrt{\dfrac{1}{D_a}}(l-x)}{\sinh\sqrt{\dfrac{1}{D_a}}l} \\ N_2 = \dfrac{\sinh\sqrt{\dfrac{1}{D_a}}x}{\sinh\sqrt{\dfrac{1}{D_a}}l} \end{cases} \tag{3.85}$$

If equation (3.80) is solved ϕ_f will be derived as:

$$\phi_f = p_d D_a (1 - (N_1 + N_2)) = p_d D_a \phi_b \tag{3.86}$$

ϕ_b is known as elemental bubble function.

3.6 Polynomial Bubble Functions

To use the bubble function expressed in equation (3.85) needs to be converted into a polynomial function from its hyperbolic form. The complexity of the hyperbolic function can only be integrated in the elemental equations manually. To change the hyperbolic functions into polynomial functions can be done by using the quadrature methods and using the Taylor series expansion. The polynomial functions are truncated after a selected number of terms and are hence derived as

$$N_1 = \frac{(l-x)\left((1+\frac{(l-x^2)}{6D_a}\right)}{l(1+\frac{1}{6D_a}h^2)} = \frac{l-x}{l} - \frac{x(l-x)(2l-x)}{l(6D_a+l^2)} \tag{3.87}$$

$$N_2 = \frac{x\left((1+\frac{1}{6D_a}x^2)\right)}{l(1+\frac{1}{6D_a}l^2)} = \frac{x}{l} - \frac{x(l-x)(l+x)}{l(6D_a+l^2)} \tag{3.88}$$

In equations (3.87) and (3.88) the second parts represent third order bubble functions :

$$\begin{cases} \phi_1 = \dfrac{x(l-x)(2l-x)}{l(6D_a+l^2)} \\ \phi_2 = \dfrac{x(l-x)(l+x)}{l(6D_a+l^2)} \end{cases} \tag{3.89}$$

Using a local coordinate system of ξ (-1,+1) the bubble functions are written as:

$$\begin{cases} \phi_1 = \dfrac{(1-\xi^2)(3-\xi)}{8\left(1+\dfrac{6D_a}{l^2}\right)} = b(3-\xi)(1-\xi^2) \\ \phi_2 = \dfrac{(1-\xi^2)(3+\xi)}{8\left(1+\dfrac{6D_a}{l^2}\right)} = b(3+\xi)(1-\xi^2) \end{cases} \tag{3.90}$$

where $\xi = 1 - \dfrac{2x}{l}$ and $b = \dfrac{1}{8\left(1+\dfrac{6D_a}{l^2}\right)}$

in which l is a characteristic element length.

Using a similar procedure fifth order bubble enriched bilinear element can also be derived as:

$$\begin{cases} \phi_1 = A[a(1-\xi^2)+b(1-\xi^2)(1-\xi)+c(1-\xi^2)^2+d(1-\xi^2)^2(1-\xi)] \\ \phi_2 = A[a(1-\xi^2)+b(1-\xi^2)(1+\xi)+c(1-\xi^2)^2+d(1-\xi^2)^2(1+\xi)] \end{cases} \tag{3.91}$$

in which

$$A=\frac{1}{l\left(1+\frac{l^2}{6D_a}+\frac{l^4}{120D_a^2}\right)}, \quad a=\left(-\frac{l}{6D_a}-\frac{l^3}{120D_a^2}\right)\frac{l^2}{4}$$

$$b=\left(-\frac{l}{6D_a}-\frac{3l^2}{120D_a^2}\right)\frac{l^3}{8}, \quad c=\frac{2l^5}{1920D_a^2}, \quad d=\frac{l^5}{3840D_a^2}$$

3.6.1 Static Condensation

When the hyperbolic functions such as equation (3.85) are approximated by the Taylor expansion, the resulting polynomial bubble functions are no longer residual free. In theory any function which is zero at the element boundary is a bubble function. The fact that bubble functions disappear on element boundaries makes it possible to remove the equations that correspond to these functions from the set of elemental equations. This procedure is called static condensation. In the residual free method, condensation takes place automatically for the derived bubble functions. Therefore, the bubble functions incorporation with Lagrangian shape functions takes place automatically. Bubble coefficients are calculated as a part of the condensation procedure. However, as an alternative, other bubble functions can be used to incorporate with linear Lagrangian shape functions by means of the static condensation procedure. Two types of bubble functions are considered in this work. With respect to the derived polynomial of the residual free method and using optional elemental polynomial bubble functions. To demonstrate this point we consider the following polynomial bubble function.

Second order elemental bubble function:

$$\phi_b = (1-\xi^2) \tag{3.92}$$

Forth order elemental bubble function:

$$\phi_b = (1-\xi^2)+(1-\xi^2)^2 \tag{3.93}$$

Sixth order elemental bubble function:

$$\phi_b = (1-\xi^2)+(1-\xi^2)^2+(1-\xi^2)^3 \tag{3.94}$$

According to the above polynomials it can be concluded that an m^th order elemental bubble function may be written as:

$$\phi_b = (1-\xi^2)+(1-\xi^2)^2+(1-\xi^2)^3+\cdots+(1-\xi^2)^m = \sum_{q=1}^{m}(1-\xi^2)^q \tag{3.95}$$

As an another choice for the bubble functions we consider this type of functions:

$$\phi_b = (1-\xi^{2n}) \qquad \text{n=1,2,3,4,....} \tag{3.96}$$

To incorporate the bubble functions with ordinary shape functions, with respect to equation (3.72) we have

$$u_h = \psi_1 u_1 + \psi_2 u_2 + \phi_b u_b \tag{3.97}$$

where ψ_i in this work is the Lagrangian linear shape function and ϕ_b is the polynomial bubble function. Using the static condensation procedure the bubble enriched one dimensional shape functions can be generally derived as:

$$N_i = \psi_i + b\phi_b \tag{3.98}$$

in which b is the bubble coefficient and is derived during the implementation of the static condensation method.

3.6.2 Derivation of two dimensional Bubble functions

Derivation of two dimensional bubble functions are obtained by using tensor products of one dimensional functions, theses are necessary for practical implementations. The derived bubble functions are then incorporated into normal interpolation functions of bilinear Lagrangian elements to obtain shape functions of a bubble enriched bilinear element as:

$$\begin{cases} N_1 = \frac{1}{4}(1-\xi)(1-\eta) - b(1-\xi^2)(1-\eta^2) \\ N_2 = \frac{1}{4}(1+\xi)(1-\eta) - b(1-\xi^2)(1-\eta^2) \\ N_3 = \frac{1}{4}(1+\xi)(1+\eta) - b(1-\xi^2)(1-\eta^2) \\ N_4 = \frac{1}{4}(1-\xi)(1+\eta) - b(1-\xi^2)(1-\eta^2) \end{cases} \quad \textbf{(3.99)}$$

Where:

$$b = \frac{1}{8\left(0.2 + \frac{2D_a}{l^2}\right)} \quad \textbf{(3.100)}$$

which is calculated during the implementation of the static condensation method. Here l is a characteristic length of the element. Using the same procedure forth order bubble enriched bilinear elements can be derived as:

$$\begin{cases} N_1 = \frac{1}{4}(1-\xi)(1-\eta) - b[(1-\xi^2)(1-\eta^2) + (1-\xi^2)^2(1-\eta^2)^2] \\ N_2 = \frac{1}{4}(1+\xi)(1-\eta) - b[(1-\xi^2)(1-\eta^2) + (1-\xi^2)^2(1-\eta^2)^2] \\ N_3 = \frac{1}{4}(1+\xi)(1+\eta) - b[(1-\xi^2)(1-\eta^2) + (1-\xi^2)^2(1-\eta^2)^2] \\ N_4 = \frac{1}{4}(1-\xi)(1+\eta) - b[(1-\xi^2)(1-\eta^2) + (1-\xi^2)^2(1-\eta^2)^2] \end{cases} \quad \textbf{(3.101)}$$

Where

$$b = \frac{1}{8\left(0.386 + \frac{3.905 D_a}{l^2}\right)}$$

Any higher order bubble enriched bilinear element can be derived similarly.

If the nth order bubble function in equation (3.96) is used the two dimensional bubble enriched bilinear shape functions can be written as:

$$\begin{cases} N_1 = \frac{1}{4}(1-\xi)(1-\eta) - b(1-\xi^{2n})(1-\eta^{2n}) \\ N_2 = \frac{1}{4}(1+\xi)(1-\eta) - b(1-\xi^{2n})(1-\eta^{2n}) \\ N_3 = \frac{1}{4}(1+\xi)(1+\eta) - b(1-\xi^{2n})(1-\eta^{2n}) \\ N_4 = \frac{1}{4}(1-\xi)(1+\eta) - b(1-\xi^{2n})(1-\eta^{2n}) \end{cases} \quad \textbf{(3.102)}$$

where b is represented as

$$b = \frac{\frac{l}{2D_a}\frac{2n}{2n+1}}{\frac{16n^2}{(4n-1)l} + \frac{8n^2 l}{(4n+1)(2n+1)D_a}}$$

3.7 Elimination of inter-element boundary integrals

When bubble functions are applied the inter-element boundary integrals are not automatically eliminated during the assembly of elemental equations. This problem does not become apparent in the one dimensional case as the boundary integrals are reduced to simple nodal flux terms. This problem can however be solved very efficiently. We consider the case of the Brinkman equation as a type of multi-scale problem. The variational formulation for the Brinkman equation, after application of Green's theorem is

$$(\frac{1}{D_a}u_h, v_1) + (\nabla u_h, \nabla v_1) = (p_d, v_1) \tag{3.103}$$

Substitution from equation (3.72) gives:

$$(\frac{1}{D_a}u_h, v_1) + (\nabla u_1, \nabla v_1) + (\nabla u_b, \nabla v_1) = (p, v_1) \tag{3.104}$$

If v_1 is a linear test function (weight function) according to Green's theorem:

$$(\nabla v_1, \nabla \phi)_{\Omega_e} = -(\Delta v_1, \phi)_{\Omega_e} + (\nabla v_1, \phi)_{\Gamma_e} = 0 \tag{3.105}$$

Where ϕ is bubble function. Therefore equation (3.104) is reduced to:

$$(\frac{1}{D_a}u_h, v_1) + (\nabla u_1, \nabla v_1) = (p, v_1) \tag{3.106}$$

As can be seen the bubble function does not affect the Laplacian term in the Brinkman equation and therefore no boundary integral due to the bubble function exists when solving this equation. The same is true for Navier-stokes equation. Although we recognise that Navier-Stokes equation foes not need necessarily represent a multi-scale problem.

The outlined approach for the development of two dimensional bubble functions is directly extendable to three dimensional problems. The three dimensional element if derived is a bubble enriched tetrahedron, it provides a more flexible element geometry for the generation of discretisation which match closely to three dimensional domains with curved boundaries. An example is shown below in figure 3.8 (Okumara and Kawahara, 2003). In this example the surface of a three dimensional object has been discretised into triangular bubble elements (figure 3.8 (a)) in which the variation of velocity is carried by a high order interpolation than the pressure as shown in figure 3.8 (b).

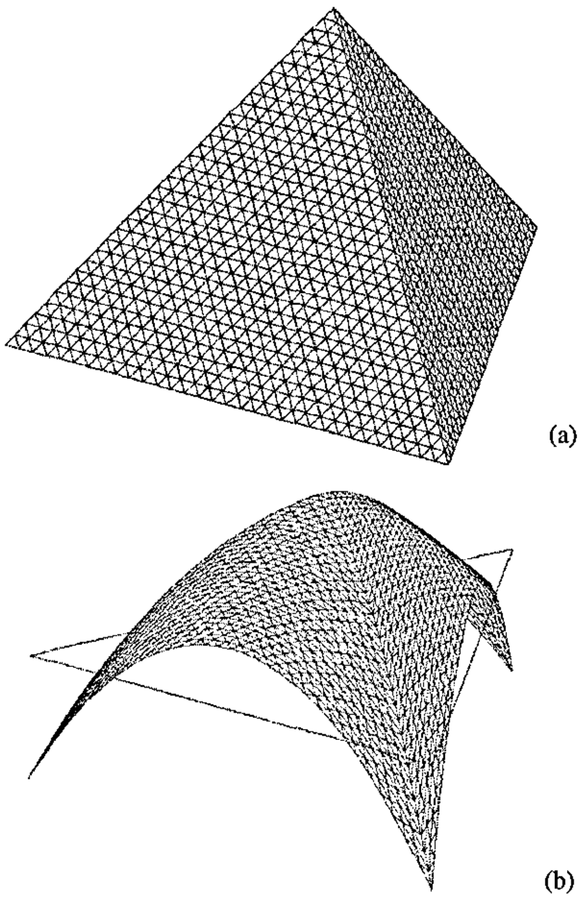


Figure 3.8 A discretisation using bubble elements and graphical representation of a bubble enriched triangular element

If the body of the object shown in figure 3.8 (a) needs to be discretised three dimensional tetrahedron element enriched with bubble function must be used.

3.8 Extension of Bubble Function approach to transient problems

The stable and accurate numerical solution of transient problems have been the subject of numerous research. In this respect variety of time-stepping techniques have been used in conjunction with finite element method (Zienkiewicz and Taylor, 1994).

However in most cases finite element approach is exclusively used for spatial discretisation and temporal treatment of the transient problem is done by alternative finite difference based simple time stepping schemes (such as theta time stepping), bubble function method can also be used to develop a systematic technique for sophisticated time dependant numerical schemes. Such schemes can be used to solve complex problems where temporal variation of the field unknowns are different at different times at different locations in a problem domain.

To illustrate the present extension we consider the case of transient convection diffusion equation for the reason that the standard Galerkin finite element solution of it is not straight forward. We recognise that SUPG (Streamline upwinding Petrov-Galerkin) can be used to solve this equation but upwinding of a transient equation is even more problematic than that of a steady state case, simply because that time-stepping itself can cause ad hoc damping of the solution.

In the case of transient convection-diffusion problems the basic issue is not only obtaining a stable approximation but also efficient coupling between the spatial and the temporal discretisations (Donea at al. 2000; Donea, 1984) in finite element schemes. It has been shown that the combination of a standard Galerkin spatial discretisation with classical time-stepping schemes such as the Lax-Wendrof, leap-frog and Crank-Nicolson methods fails to produce satisfactory numerical results when the transport process is convection dominated. This is because that time integration methods are only effective if very small time steps are used, and this severely undermines the utility of such schemes in practical applications (Donea et al., 2000). Schemes involving first order time derivatives are indeed easier to implement for solving unsteady convection-diffusion problems than, for instance, the standard third- and fourth- order accurate Taylor-Galerkin schemes which imply the substitution of the higher order time derivatives with spatial derivatives (Donea et al., 1984). Another difficulty stemming from the complexity of using higher order temporal discretisation such as 3rd or 4th order Taylor-Galerkin schemes. Therefore for implementation with C_0 finite elements, higher order time-stepping schemes for the convection-diffusion equation should not be extended to involve higher-order time-stepping derivatives. This has been the reason for the construction of multi-stage schemes emanating from Pade approximations to the exponential function (Argyris et al., 1997) as well as the Runge-Kutta methods (Lambert, 1993). In these methods the computation cost of transient time-stepping algorithm allows us to use larger time steps (Donea et al., 2000).

In this section we develop a scheme which resolves many of the difficulties regarding transient problems. In addition using this approach the multi-scale finite element modelling for time approximation becomes possible. Multi-scale approach is used for those problems in which, capturing for all physical phenomenon needs a high level of discretisation if an accurate simulation is required. Multi-scale finite element modelling without mesh refinement (Parvazinia et al., 2006a; 2006b).we show that a simpler approach can be used to obtain accurate solution for transient problems without the use of complex time stepping. The idea of using finite element discretisation for time variables have been first proposed more than two decades ago (Gratlop). However, these early attempts were abandoned because they didn't address the real problem of multi-scale transient dynamics. Theoretically removing the need for excessive mesh refinement, the variational multi-scale method provides a framework for separating the treatment of terms having different scales provides a framework for separating the treatment of terms having different scales. For a two-scale method, the field unknown is divided into two parts as $T = T_l + T_b$, where T_b is called fine, sub-grid or unresolved scale and maybe derived analytically while T_l is called coarse or unresolved scale and is represented by the standard polynomial finite element approximation. A method for generating practical multi-scale schemes is based on the use of bubble enhance trial functions in a finite element discretisation. Bubble function are, typically, high order polynomials which vanish on the element boundaries (Brezzi et al., 1992 and 1997; Baoicchi et al., 1993; Franca et al., 1993 ; franca et al, 1997; Franca and Russo, 1996; 1997). These functions can be used to enrich the ordinary linear Lagrangian elements to generate higher order approximations without increasing the order of the elements in the nominal sense.

In the present study the multi-scale finite element discretisation based on the bubble function is used for temporal approximation of time dependent variables. For the convection-diffusion equation the multi-scale modelling is used in both spatial and temporal approximations. The results, in comparison with the analytical solution of a bench mark problem, show that the proposed scheme is capable of yielding accurate and stable results. In addition, the results of the proposed method are compared with widely used theta time stepping method, to illustrate the performance of the proposed scheme.

3.9 Multi-scale Finite element modelling for transient case

For transient convection-diffusion equation multi-scale behaviour in both temporal and spatial variables may be present. For stable solution of such problems bubble functions can be used to resolve the difficulty of excessive mesh refinement. Bubble functions for steady transport equations are previously by Parvazinia et al. (2006a, 2006b) and will not be given, however a summary is given in the next section to eliminate the extension of technique to transient problems.

3.9.1 Incorporation of the Bubble Function

Let us consider a boundary value problem defined in $\Omega \subset R^2$ as

$$\begin{cases} LT = f & in\Omega \\ T = 0 & on\Gamma \end{cases} \tag{3.107}$$

where L is a linear differential operator and *f* is a given source function defined on *Ω*. The standard Galerkin method is formulated in a subspace $V_h \subset V$, where V is the space of functions for which a solution of the continuous problem is sought. The Galerkin method aims to find $u_h \subset V_h$ such that

$$a(T_h, v) = (LT_h, v) = (f, v) \tag{3.108}$$

Where a(. , .) is a bilinear for and a(. , .) representing the scalar product of its arguments. In a two-scale method, the unknowns are divided into two parts

$$\begin{cases} T_h = T_1 + T_b \\ v_h = v_1 + v_b \end{cases} \tag{3.109}$$

Where T_1 *is the fine scale and* T_b *represents a standard finite element finite element* approximation polynomial (interpolation function. Since the bubble functions disappear on element boundaries makes it possible to remove the equations that correspond to these functions from the set of elemental equations. This process is called static condensation (Bathe, 1996) using the static condensation we set $v = v_b$ in equation (3.108) to obtain the following variational formulation (Hughes, 1995)

$$a(v_h, T_h) = (v_h, f) \text{ or } a(v_1 + v_b, T_1 + T_b) = (v_1 + v_b, f) \tag{3.110}$$

Statement (3.112) can be written as two sub-problems as

$$\{a(v_b, T_1) + a(v_b, T_b) = (v_b, f) \tag{3.111}$$

$$\{a(v_1, T_1) + a(v_1, T_b) = (v_1, f) \tag{3.112}$$

In a steady state problem the behaviour is the same in all directions and elemental bubble is applied using the same with coefficient in all direction. In a constant transient problem the spatial and temporal behaviour are usually different using a higher elemental bubble function may become impractical. This can usually be solved by separating the spatial and temporal bubble functions as

$$\begin{cases} T_b = T_{bs} + T_{bt} \\ v_b = v_{bs} + v_{bt} \end{cases} \tag{3.113}$$

Therefore equations (3.112) can be rewritten as

$$a(v_{bs} + v_{bt}, T_1) + a(v_{bs} + v_{bt}, T_{bs} + T_{bt}) = (v_{bs} + v_{bt}, f) \tag{3.114}$$

Again in terms of two sub-problems as

$$\{a(v_{bs},T_1)+a(v_{bs},T_{bs})+a(v_{bs}+T_{bt})=(v_{bs},f) \tag{3.115}$$

$$\{a(v_{bt},T_1)+a(v_{bt},T_{bs})+a(v_{bt}+T_{bt})=(v_{bt},f) \tag{3.116}$$

Equation (3.117) and either Equation (3.118) or Equation (3.119) imply that separate static condensation for spatial and time directions have been used to solve problems.

Using a 2nd order bubble function in x and 4th order bubble function in t the bubble enriched Lagrangian shape function in local coordinate system ξ(-1,+1),τ(-1,+1) can be written as

$$\begin{cases} N_1=\frac{1}{4}(1-\xi)(1-\tau)+b(1-\xi^2)+bt\phi_\tau \\ N_2=\frac{1}{4}(1+\xi)(1-\tau)+b(1-\xi^2)+bt\phi_\tau \\ N_3=\frac{1}{4}(1+\xi)(1+\tau)+b(1-\xi^2)+bt\phi_\tau \\ N_4=\frac{1}{4}(1-\xi)(1+\tau)+b(1-\xi^2)+bt\phi_\tau \end{cases} \tag{3.117}$$

where ϕ_τ is the 4th order bubble function for time b and b_t are bubble coefficients in x and t directions, respectively. Two types of time dependant bubble functions are used

$$\left\{\phi_\tau=\sum_{q=1}^{n}(1-\tau^2)^q\right. \tag{3.118}$$

$$\left\{\phi_\tau=\sum_{q=1}^{n}(1-\tau^{2q})\right. \tag{3.119}$$

for q=1 the bubble function is 2nd order, as q=2 the bubble function is 4th order, and so on.

The derived bubble function are used to solve a typical transient problem, namely transient convection-diffusion problem. The details of this solution are given in chapter 4 of this thesis.

Chapter 4
Computational Results and Discussion

4.1 Introduction

In this chapter we present a number of simulations that show the applicability and performance of the bubble function method. The first set of results relate to the solution of a multi-scale porous flow problem. The second set of results represent the extension of the method to transient flow problem. The final part of this chapter consists of the simulation of a realistic problem, namely salt intrusion in the Upper Milford-Haven Estuary and comparison of simulation results with experimentally collected field survey data.

4.2 Bench Mark Problem 1

In a porous medium, flow can be represented by different types of governing equations depending on the range of the permeability of the domain and the flow Reynolds number. In highly permeable porous media, low Reynolds number flow regimes can be represented by the Brinkman equation. In this type of flow where permeability of porous matrix is high the fluid carries some of the imposed stress. This effect rises sharply in near wall layers as the permeability of porous media decreases. It is interpreted as the flow system having different scales, a 'fine scale' in the near wall zone and a 'coarse scale' in the rest of the domain. Therefore, theoretically accurate modelling of the Brinkman regime can only be obtained via excessive mesh refinement of the solution domain, at least in the region of the boundary layer. However, the thickness of the boundary layer is not known a priori and depends on the domain permeability. This in turn makes the classical schemes such as the standard Galerkin method unsuitable for a multi-scale problem such as the Brinkman equation. These types of problems can be modelled using multi-scale variational methods. These techniques are currently used to solve problems related to turbulent flows, structural analysis of composite materials, flow through porous media, weather forecasting and large-scale molecular dynamic simulations. Representation of all physical scales need a high level of discretisation which is a common difficulty with these problems. To have stable-accurate solution, the multi-scale method should be capable of incorporating the influence of the fine-scales while using discretisation at a coarse level to avoid excessive mesh refinement.

In a two-scale method, the field unknown is divided into two parts as $u = u_1 + u_b$, where u_b is known as fine, sub-grid or unresolved scale which may be derived analytically and u_1 is known as coarse or resolved scale where is represented by a standard polynomial finite element approximation. In spite of theoretical progresses in this area the development of algorithms which enable implementation of the theoretical considerations in practice is not a trivial matter. Bubble functions can be incorporated in a finite element discretisation to generate a multi-scale scheme. These functions are, typically, high order polynomials which vanish on the element boundaries. The bubble functions can be systematically derived using the residual free bubble method. The essential idea of this method is that the bubble functions should satisfy, strongly, the model differential equation within each element subject to homogeneous boundary conditions. In multi-dimensional problems, the analytical solution of a partial differential equation (PDE) in the residual free method within each element is a major task. The analytical solution of a PDE can be replaced by the analytical solution of an ODE (ordinary differential equation), in the residual free bubble function method. To this end the exact solution of the ODE is approximated by the Taylor series expansion and the multi-dimensional bubble functions are derived by tensor product of one dimensional bubbles.

A continuous penalty scheme is used to evaluate polynomial bubble functions in multi-scale finite element solution of the flow in porous media with curved and contracting boundaries using the Brinkman equation. The method of incorporating bubble functions with Lagrangian shape functions using static condensation method, derivation of two dimensional bubble functions and elimination of the boundary integrals are explained. The numerical results are validated with analytical solution in a simple rectangular domain and then the isothermal flow of a Newtonian fluid is studied in different domains

4.2.1 Governing Equations

The governing equations of isothermal flow of Newtonian fluids in a porous duct with impermeable walls (Figure 4.1) in a two dimensional Cartesian coordinate system are given by:

Continuity equation:

$$\frac{\partial u}{\partial \mathrm{x}}+\frac{\partial v}{\partial \mathrm{y}}=0 \tag{4.1}$$

x-component of the Brinkman equation:

$$-\frac{\partial p}{\partial x}-\frac{\mu}{K}u+\mu_e\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=0 \tag{4.2}$$

y-component of the Brinkman equation:

$$-\frac{\partial p}{\partial y}-\frac{\mu}{K}v+\mu_e\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)=0 \qquad (4.3)$$

Where u and v are velocity components, p is pressure, μ is fluid viscosity, K is the domain permeability and μ_e is the effective viscosity that theoretically takes into account the stress borne by the fluid as it flows through a porous medium. However, experimental measurement of μ_e is not a trivial matter, if not impossible. Therefore, in the present work in accordance with overwhelming majority of the published literatures μ_e is set to be equal to the fluid viscosity μ.

4.2.2 Boundary Conditions

We use the following boundary conditions (see Figure 4.1):

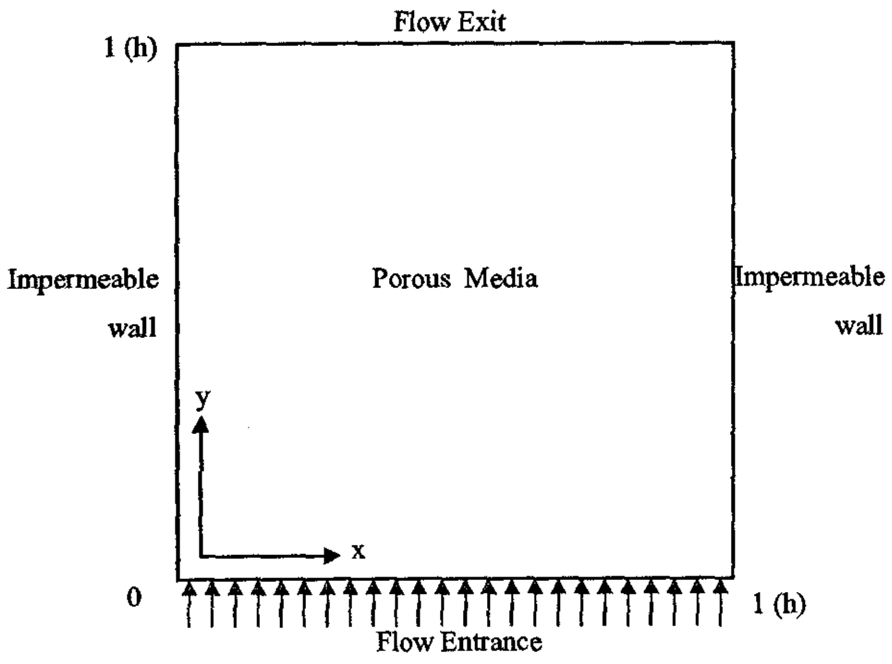


Figure 4.1 Rectangular flow domain (domain 1) and boundaries used for model validation

I) Inlet to the domain

In accordance with majority of engineering flow processes at the inlet a plug flow condition is applied. This can be written as follows:

$$u = 0,\ v = v_0 \quad \text{for } 0 < x < h \text{ and } y = 0 \tag{4.4}$$

where h is the gap width in a rectangular domain.

II) At impermeable (solid) walls

$$\begin{array}{lll} u = 0, v = 0 & \text{for} & x = 0 \text{ and } 0 \le y < h \\ u = 0, v = 0 & \text{for} & x = h \text{ and } 0 \le y < h \end{array} \tag{4.5}$$

III) Exit

At the outlet a stress free condition is used, therefore both shear and normal components of the surface forces are set to zero.

$$\tau_{yy}\big|_{exit} = 2\mu\frac{\partial v}{\partial y} = 0 \qquad \text{for } y = h \text{ and } 0 \le x \le h \tag{4.6}$$

$$\tau_{yx}\big|_{exit} = \tau_{xy}\big|_{exit} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) = 0 \qquad \text{for } y = h \text{ and } 0 \le x \le h \tag{4.7}$$

$$\tau_{xx}\big|_{exit} = 2\mu\frac{\partial u}{\partial x} = 0 \qquad \text{for } y = h \text{ and } 0 \le x \le h \tag{4.8}$$

The use of 'stress free' instead of 'developed flow' conditions provides a more general exit boundary condition enabling the simulation of realistic situations where the flow development cannot be guaranteed.

4.2.3 Dimensionless form of Governing Equations

To preserve the consistency of the numerical solutions we use the following dimensionless variables:

$$y^* = \frac{y}{h}, x^* = \frac{x}{h}, u^* = \frac{u\mu}{\rho g h^2}, v^* = \frac{v\mu}{\rho g h^2}, p^* = \frac{p}{\rho g h}$$

$$\tau_{xx}^* = \frac{\tau_{xx}}{\rho g h}, \tau_{yy}^* = \frac{\tau_{yy}}{\rho g h}, \tau_{yx}^* = \frac{\tau_{yx}}{\rho g h}, \tau_{xy}^* = \frac{\tau_{xy}}{\rho g h}$$

Where ρ is the fluid density, p is the pressure and g is acceleration due to gravity. Substituting the defined dimensionless variables in Equations (4.1to 4.8) the following dimensionless governing equations are obtained:

$$\frac{\partial u^*}{\partial x^*} + \frac{\partial v^*}{\partial y^*} = 0 \tag{4.9}$$

$$-\frac{\partial p^*}{\partial x^*} - \frac{1}{Da} u^* + \left(\frac{\partial^2 u^*}{\partial x^{*2}} + \frac{\partial^2 u^*}{\partial y^{*2}} \right) = 0 \tag{4.10}$$

$$-\frac{\partial p^*}{\partial y^*} - \frac{1}{Da} v^* + \left(\frac{\partial^2 v^*}{\partial x^{*2}} + \frac{\partial^2 v^*}{\partial y^{*2}} \right) = 0 \tag{4.11}$$

Where Da is the Darcy parameter defined as:

$Da=K/h^2$

The corresponding dimensionless boundary conditions are expressed as:

I) Entrance

$$u^* = 0 \quad , \quad v^* = v_0^* \qquad \text{for } y=0 \text{ and } 0<x<1 \tag{4.12}$$

In this work v_0^* was selected to be equal to 0.01. This is to assure that the flow regime remains laminar and the inertia term can be neglected.

II) Impermeable walls

$$\begin{aligned} &u^* = v^* = 0 \qquad \text{for } x^* = 0 \text{ and } 0 \le y^* < 1 \\ &u^* = v^* = 0 \qquad \text{for } x^* = 1 \text{ and } 0 \le y^* < 1 \end{aligned} \qquad \textbf{(4.13)}$$

III) Exit- Stress free conditions expressed in the dimensionless form are imposed.

$$\tau_{yy}^* = 2\frac{\partial v^*}{\partial y^*} = 0 \qquad \text{for } y^* = 1 \text{ and } 0 \le x^* \le 1 \qquad \textbf{(4.14)}$$

$$\tau_{yx}^* = \tau_{xy}^* = \left(\frac{\partial u^*}{\partial y^*} + \frac{\partial v^*}{\partial x^*}\right) = 0 \qquad \text{for } y^* = 1 \text{ and } 0 \le x^* \le 1 \qquad \textbf{(4.15)}$$

$$\tau_{xx}^* = 2\frac{\partial u^*}{\partial x^*} = 0 \qquad \text{for } y^* = 1 \text{ and } 0 \le x^* \le 1 \qquad \textbf{(4.16)}$$

4.2.4 Finite Element Scheme

There are a variety of different finite element schemes that can be used for the solution of the governing equations of porous flow regimes. The finite element scheme used in the present work is based on the continuous penalty technique. This technique is in essence similar to the 'Lagrange Multiplier Method' used for the solution of differential equations subject to a constraint. Here the continuity equation (i.e. the incompressibility condition) is regarded as a constraint for the equation of motion. Therefore instead of solving the governing flow equations as a system of three P.D.E. s the pressure in the components of the equation of motion is replaced by a multiplier (called penalty parameter) times the continuity equation. This gives a more compact set of working equations with components of the velocity as the remaining unknowns. Additionally, elimination of the pressure from the equation of motion automatically satisfies the basic numerical stability condition for the simulation of incompressible flows, known as the LBB criteria. The mathematical theory underpinning the development of LBB criterion is somewhat obscure. However, it can be readily observed that the absence of a pressure term in the incompressible continuity equation makes the possibility of a mismatch between approximations used to satisfy the equations of motion and continuity almost inevitable in any numerical solution of a system of P.D.E s with velocity and pressure as the prime unknowns. It has been proved that for the bubble enriched bilinear elements the LBB condition is still satisfied. If the continuous penalty scheme is used, after representing the unknowns based on the trial functions the governing equations can be written as:

$$\int_{\Omega_e} W_i [\frac{\partial}{\partial x^*}\lambda_0(\frac{\partial\sum_{j=1}^{n} N_j u^*_j}{\partial x^*} + \frac{\partial\sum_{j=1}^{n} N_j v^*_j}{\partial y^*}) - \frac{1}{Da}\sum_{J=1}^{n} N_j u^*_j + (\frac{\partial^2\sum_{j=1}^{n} N_j u^*_j}{\partial x^{*2}} + \frac{\partial^2\sum_{j=1}^{n} N_j u^*_j}{\partial y^{*2}})] dx^* dy^* = 0$$

$$\int_{\Omega_e} W_i [\frac{\partial}{\partial y^*}\lambda_0(\frac{\partial\sum_{j=1}^{n} N_j u^*_j}{\partial x^*} + \frac{\partial\sum_{j=1}^{n} N_j v^*_j}{\partial y^*}) - \frac{1}{Da}\sum_{J=1}^{n} N_j v^*_j + (\frac{\partial^2\sum_{j=1}^{n} N_j v^*_j}{\partial x^{*2}} + \frac{\partial^2\sum_{j=1}^{n} N_j v^*_j}{\partial y^{*2}})] dx^* dy^* = 0$$

where W_i is a weight function and is equal to the Lagrangian shape function ψ_i in the standard Galerkin method and N_j is the bubble enriched shape function. Corresponding to a total of n interpolation functions, n equations are generated and a system of $n \times n$ equations is constructed. Using matrix notation this system is written as:

$$\begin{bmatrix} A_{ij}^{11} & A_{ij}^{12} \\ A_{ij}^{21} & A_{ij}^{22} \end{bmatrix} \begin{Bmatrix} u^*_j \\ v^*_j \end{Bmatrix} = \begin{Bmatrix} B_j^1 \\ B_j^2 \end{Bmatrix} \qquad \textbf{(4.17)}$$

Where

$$A_{ij}^{11} = \int_{\Omega_e} \left[(\lambda_0 + 1)\left(\frac{\partial W_i}{\partial x^*} \frac{\partial N_j}{\partial x^*} \right) + \frac{\partial W_i}{\partial y^*} \frac{\partial N_j}{\partial y^*} + \frac{1}{D_a} W_i N_j \right] dx^* dy^*$$

$$A_{ij}^{12} = \int_{\Omega_e} \lambda_0 \frac{\partial W_i}{\partial x^*} \frac{\partial N_j}{\partial y^*} dx^* dy^*$$

$$A_{ij}^{21} = \int_{\Omega_e} \lambda_0 \frac{\partial W_i}{\partial y^*} \frac{\partial N_j}{\partial x^*} dx^* dy^*$$

$$A_{ij}^{22} = \int_{\Omega_e} \left[(\lambda_0 + 1)\left(\frac{\partial W_i}{\partial y^*} \frac{\partial N_j}{\partial y^*} \right) + \frac{\partial W_i}{\partial x^*} \frac{\partial N_j}{\partial x^*} + \frac{1}{Da} W_i N_j \right] dx^* dy^*$$

$$B_j^2 = \int_{\Gamma_e} W_i \left\{ \left(\frac{\partial v^{*e}}{\partial x^*} \right) n_x + \left[\lambda_0 \left(\frac{\partial u^{*e}}{\partial x^*} + \frac{\partial v^{*e}}{\partial y^*} \right) + \frac{\partial v^{*e}}{\partial y^*} \right] n_y \right\} d\Gamma_e$$

A system of weighted residual equations should be derived for each element in the domain. This is obviously not convenient. However, by using an elemental coordinate system rather than the global coordinates the uniformity of the matrix Equation (4.17) can be preserved. This is achieved via using isoparametric mapping of elements of the global mesh into a master element where all the calculations are carried out. In addition, a natural coordinate system such as $-1 \leq \xi, \eta \geq +1$can be used within the master element to enable the evaluation of all integrals within its domain by Gauss quadrature method.

Repeated application of the above procedure to each element in the computational mesh leads to the construction of elemental weighted residual equations written in matrix notations. Subsequent assembly of these equations over the common nodes between elements provides a system of global algebraic equations. Imposition of all boundary conditions into the assembled set of working equations renders the global system determinate which is then solved using a solution technique such as the Gaussian elimination method. A computationally efficient version of this method which relies on bit by bit reducing of the global system to upper triangular form according to an advancing front is used in the present work.

4.2.5 Analytical Solution

To evaluate the accuracy of the numerical solutions obtained using bubble enriched elements they are compared with an analytical solution. The dimensionless Brinkman equation in one dimension corresponding to a constant pressure drop can be written as:

$$\begin{cases} \dfrac{d^2v^*}{dx^{*2}} - \dfrac{1}{D_a}v^* + p_d^* = 0 \\ v^* = 0 \qquad \text{at } x^* = 0 \\ v^* = 0 \qquad \text{at } x^* = 1 \end{cases}$$

where

$$p_d^* = -\frac{\partial P^*}{\partial y^*}$$

Solution of the above equation gives

$$v^* = \frac{p_d^* D_a (e^{-\alpha} - 1)}{(e^{\alpha} - e^{-\alpha})}(e^{\alpha x^*} - e^{-\alpha x^*}) + p_d^* D_a (1 - e^{-\alpha x^*})$$

where:

$$\alpha = \sqrt{\frac{1}{D_a}}$$

To calculate pressure so that the consistency of the solution with numerical results is preserved, the average velocity has to be equal to the input plug flow velocity. In the present domain it is written as:

$$\int_0^1 v^* dx^* = v_0^*$$

Solution of the above equation gives:

$$p_d^* = \frac{1}{D_a} v_0^* \left(1 + \frac{1}{\alpha}\left(e^{-\alpha} - 1\right)\left(1 + \frac{D_a(e^{\alpha} - e^{-\alpha} - 2)}{e^{\alpha} - e^{-\alpha}}\right)\right)^{-1}$$

Excess pressure loss due to the entrance region can be neglected.

4.2.6 Results

To investigate the effect of using bubble enriched finite elements we have conducted a series of numerical experiments. These experiments cover a wide range of the Darcy parameter (permeability) using both ordinary Lagrangian elements and bubble enriched elements. The performance of different types of bubble functions are studied. In order to be able to use bubble enriched elements flexibly, an in-house developed computer code in FORTRAN was used to carry out the finite element simulations. In all of the presented simulations 4-noded Lagrangian elements are used. Three different domains are used to evaluate the developed multi-scale method. A rectangular domain with constant mesh density (30X30 rectangular elements) for numerical model validation (domain 1, figure 4.1), a contracting and expanding domain with curvilinear boundaries (30X60 mesh density) to study the effects of deviation from the most simple geometry and element (domain 2, figure 4.2) and a sudden contracting domain (domain 3, figure 4.3) with constant mesh density similar to domain 1.

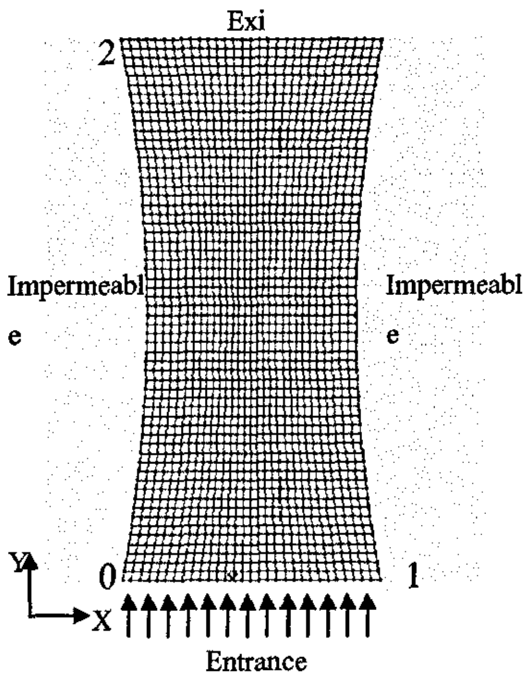


Figure 4.2 Flow domain and boundaries with variable cross section width and curved sides (domain 2).

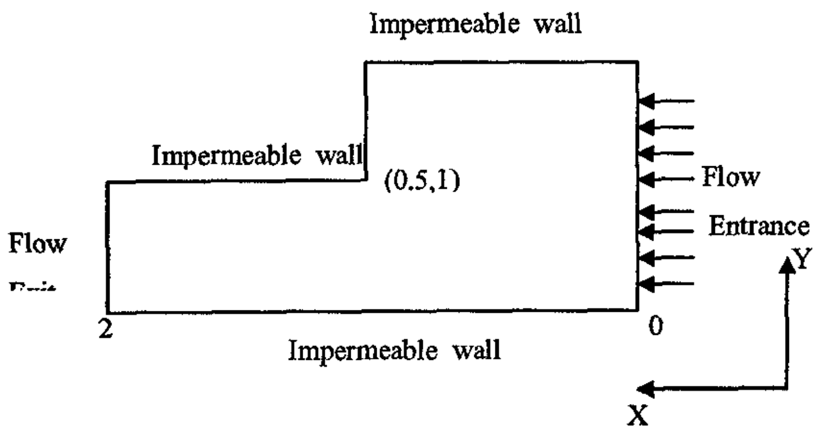


Figure 4.3 Sudden contraction flow domain (domain 3)

The first series of experiments are performed in domain 1 to compare the results of different types of bubble functions and evaluate the accuracy with respect to analytical solution. The comparison of the bubble function is carried out on this domain and on the other domains only the bubble functions which are represented in Equations (3.99) and (3.101) are evaluated. Figure 4.1 shows the domain and its boundaries. Figure 4.2 demonstrates a comparison between the 3rd and 5th order bubble functions derived directly from residual free method at Da=10-5. As Figure 4.4 shows ordinary elements fail to generate a stable and accurate solution while bubble enriched elements give stable and, in comparison with analytical solution, accurate results. The accuracy increases when 5th order bubble is used.

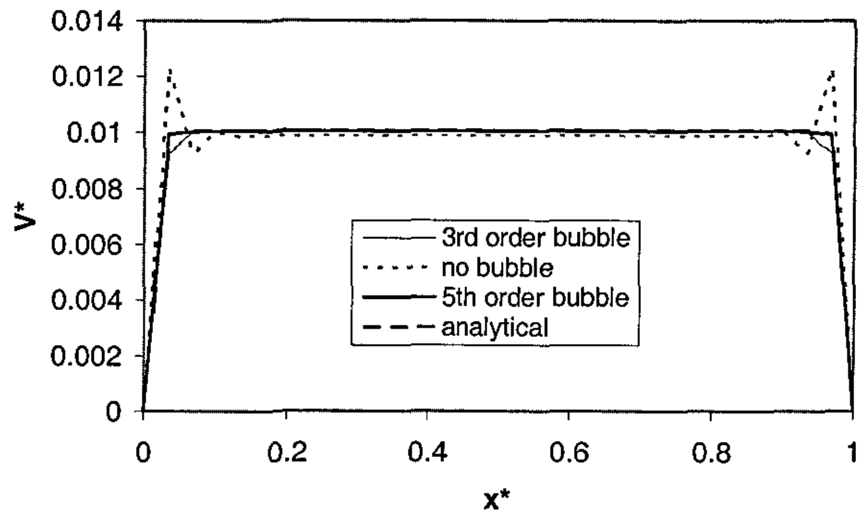


Figure 4.4 Comparison of the numerical and analytical velocity profiles at mid-height cross section for Da=10^{-5}.

At other Darcy parameters the same trend is observed , however, to avoid repetition they are not shown here. Figure 4.5 shows the results for 2nd and 4th order bubble functions which are represented in Equations (3.99) and (3.101). It is seen that the same results as figure 4.4 is achieved and by increasing the order of the bubble function the accuracy of the numerical solution increases.

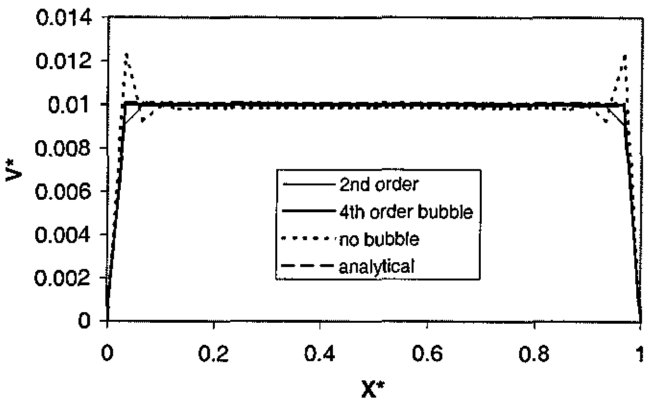


Figure 4.5 Velocity profile at mid-height cross section and Da=10^{-5} for domain 1.

The results for the bubble functions in Equation (3.102) are represented in figure 4.6. As the results show by increasing the degree of the bubble function the accuracy of the numerical solution decreases while the solution is stable. These results show that although theoretically any bubble function has stabilizing effect on the solution but it is not necessarily accurate.

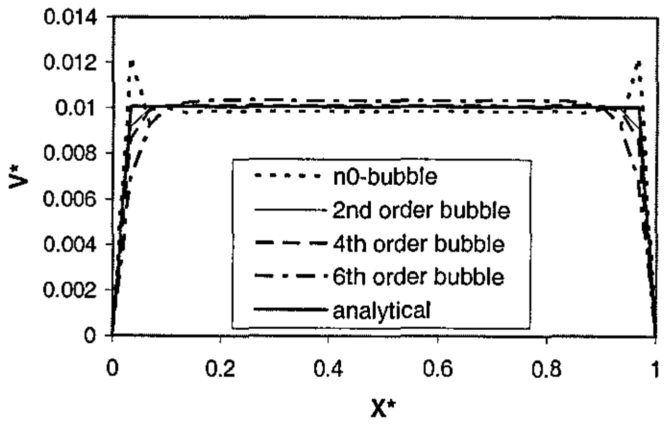


Figure 4.6 Velocity profile comparison at mid-height cross section and Da=10^{-5} for domain 1 using the bubble functions of the form $\phi_b = (1-\xi^{2n})(1-\eta^{2n})$.

Figures 4.7 to 4.11 show the results for dimensionless velocity at different cross sections of the domain 2 (figure 4.2). In this domain the boundary conditions are similar to domain 1 (figure 4.1), i.e. plug flow at the inlet, no-slip boundary conditions at the solid walls and stress free conditions at the outlet of the domain. Figure 4.7 shows the result of 2nd order bubble function at y*=0.667 and $Da=10^{-4}$. It is obvious that the bubble enriched element gives a stable solution.

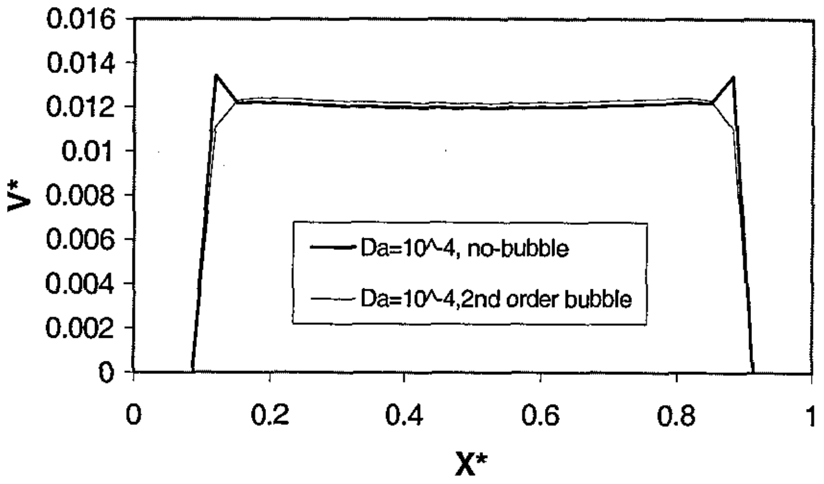


Figure 4.7 Dimensionless velocity at the cross section y=0.667 (domain 2) for $Da=10^{-4}$ and 2nd order bubble function.

Figure 4.8 shows the same result at y*=1.33. Therefore, for both contracting and expanding sections with curvilinear boundaries the bubble enriched elements give stable results. This implies that the mapping error between the quadrilateral elements with curved sides into master element has no deteriorating effect on the performance of the developed method.

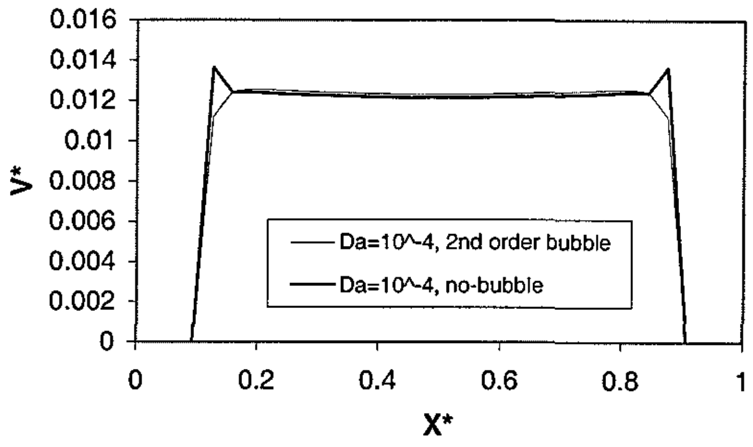


Figure 4.8 dimensionless velocity at the cross section y=1.33 (domain 2) for Da=10^{-4} and 2nd order bubble function.

Figures 4.9 to 4.11 show the results at y*=1 and different Darcy parameters. By decreasing the Darcy parameter, instability increases for ordinary elements, but using bubble functions the solution remains stable and accurate. As expected, the 4th order bubble function gives more accurate results in comparison with the 2nd order bubble function.

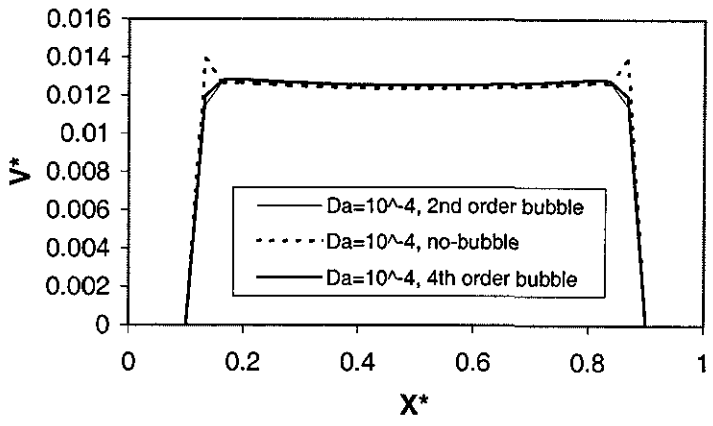


Figure 4.9 Dimensionless velocity at the cross section y=1 (domain 2). Comparison at Da=10^{-4} for 2nd and 4th order bubble functions.

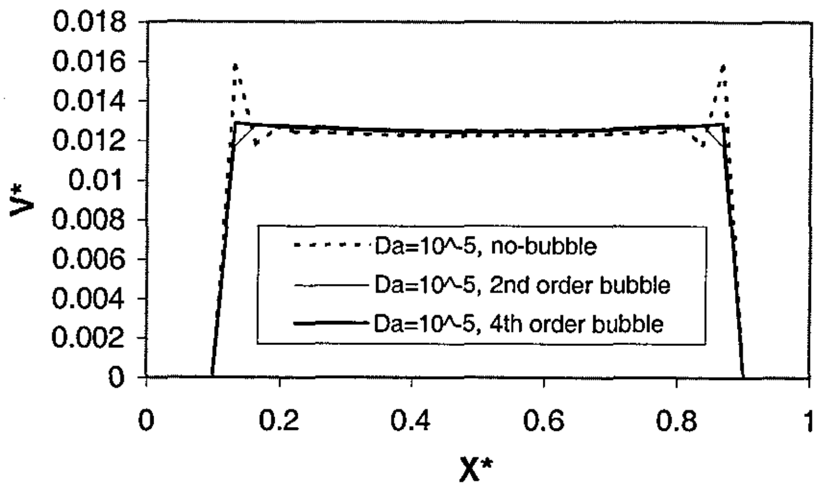


Figure 4.10 Dimensionless velocity at the cross section y=1 (domain 2). Comparison at D=10^{-5} for 2nd order and 4th order bubble function.

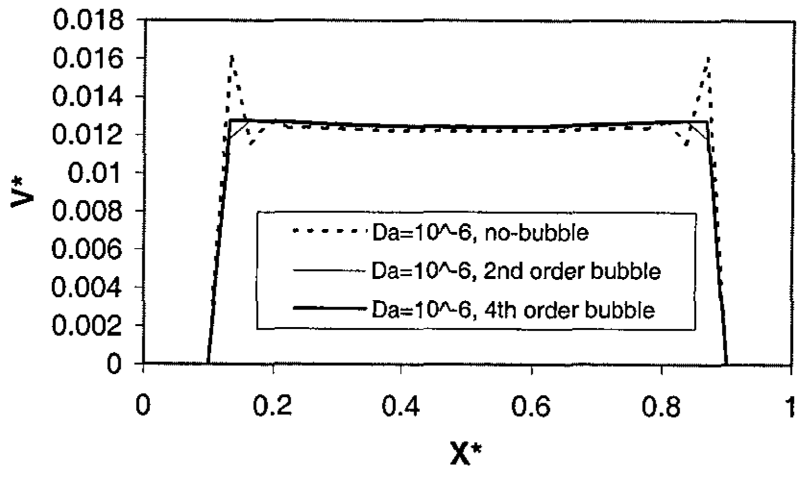


Figure 4.11 Dimensionless velocity at the cross section y=1 (domain 2). Comparison at Da=10^{-6} for 2nd order and 4th order bubble functions.

Figure 4.12 shows the dimensionless pressure field in domain 2 which matches the theoretically expected result.

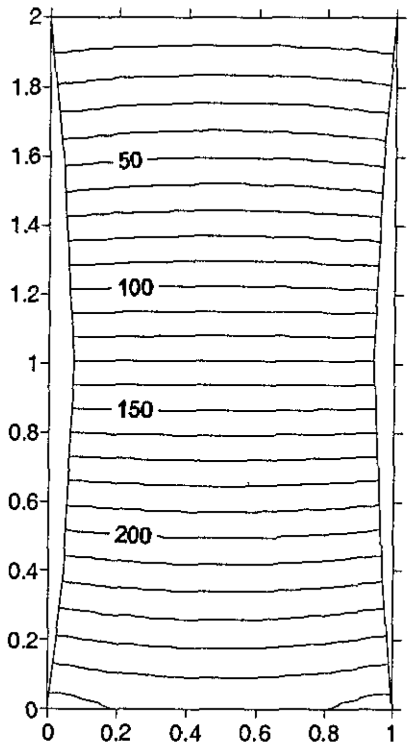


Figure 4.12 Dimensionless pressure field for Da=10^{-4} in domain 2.

An abruptly contracting domain, which includes a point of singularity, is also considered. Figure 4.3 shows the domain and its boundaries discretised using the same mesh density as the rectangular domain shown in figure 4.1. The ratio of contraction is 2/1. Simulations are based on using the same boundary conditions as for the rectangular domain. Figure 4.13 shows the 2D image of the flow field obtained using 2nd order bubble functions.

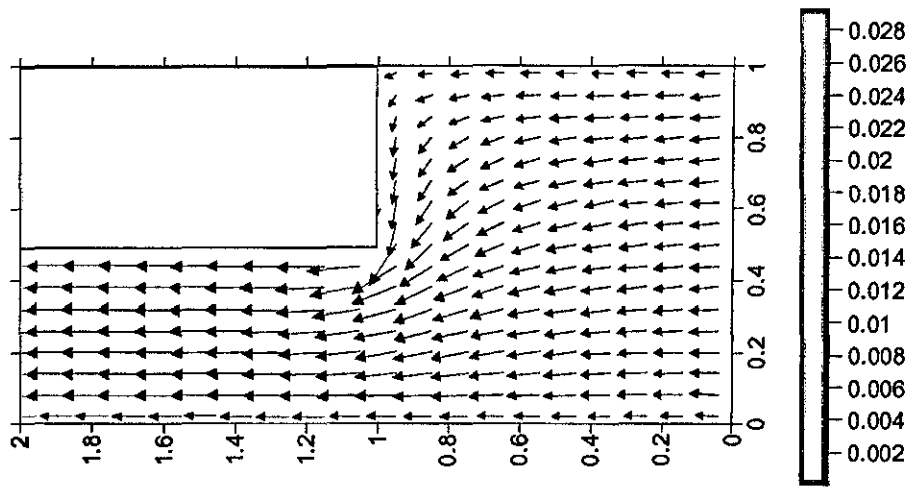


Figure 4.13 2Dimensional image of the flow field for domain 3.

Figure 4.14 represents the dimensionless velocity at the cross section corresponding to $y^*=0.5$ and $Da=10^{-4}$. Using $Da=10^{-5}$, as figure 4.15 shows, some slight instability through the whole cross section is observed. These oscillations are eliminated using 2nd order bubble enriched elements.

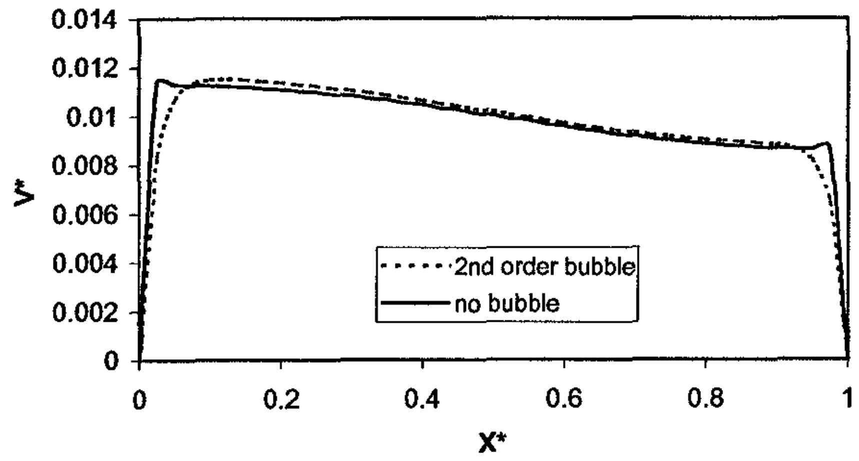


Figure 4.14 Dimensionless velocity profile comparison cross section y=0.5 and Da=10^{-4} for domain 3.

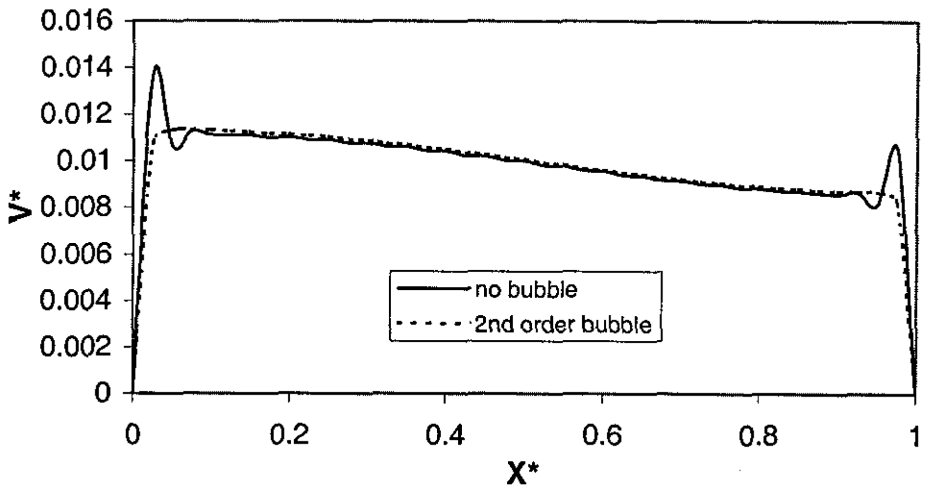


Figure 4.15 Dimensionless velocity profile comparison cross section y=0.5 and Da=10^{-5} for domain 3.

The pressure field corresponding to Da=10^{-4} is illustrated in figure 4.16 which matches the theoretically expected result.

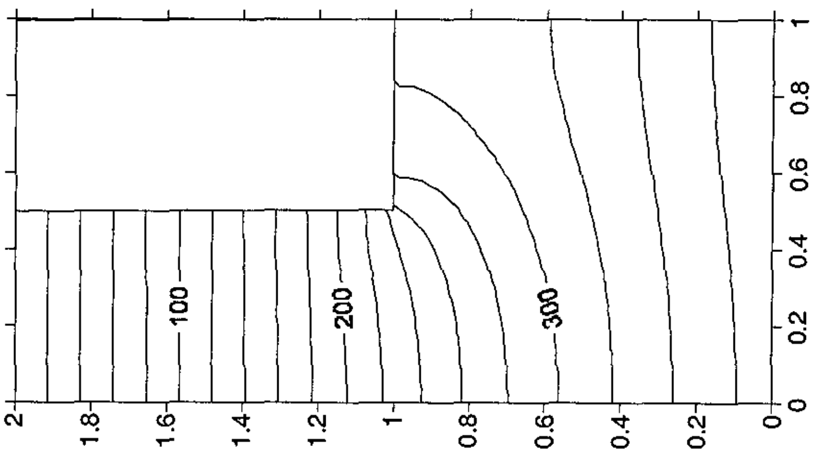


Figure 4.16 2 Dimensional image of the dimensionless pressure field at Da=10^{-4} for domain 3.

4.2.7 Conclusions

Different types of bubble functions are evaluated in the simulation of flow in highly permeable porous media using the Brinkman equation. The derivation of two dimensional bubble enriched shape functions, their implementation and performances are presented. Static condensation method were used to incorporate the bubble functions with the ordinary shape functions to perform a multi-scale finite element scheme. The numerical results show that those bubble functions where are derived directly by residual free method or with the same structure yield stable and accurate solution. For other kind of the bubble function the solution is stable but the level of accuracy can not be guaranteed. The successful bubble functions were used to other domains rather than the simple rectangular domain. Discretisations using bubble enriched elements are shown to generate stable accurate simulations for domains involving curved boundaries and abrupt changes of geometry. Although the presented method was used to model the flow in highly permeable porous media it should be considered as a general technique for multi-scale finite element solution of transport phenomena involving multi-scale behaviour.

4.3 Bench Mark Problem 2

To validate the numerical solution, the analytical solution of the dimensionless equation is present. The solution is based on the Laplace transform method.

Transient diffusion problem

$$T^*\left(x^*,t^*\right)=\frac{2}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^n}{n}\exp\left(\frac{-n^2\pi^2Dt^*}{l}\right)\sin\left(\frac{n\pi x^*}{l}\right)+\frac{x^*}{l} \qquad \textbf{(4.18)}$$

transient convection-diffusion problem for D=1 (in all simulation D=1 and C is adjusted).

$$T^*(x^*,t^*) = \exp\left(0.5\frac{C}{D}(x^*-1)\right)\sum_{n=1}^{\infty}\frac{2\pi}{n^2\pi^2+0.25\left(\frac{C}{D}\right)^2}n(-1)^n$$

$$\exp\left(\frac{-\left(n^2\pi^2+0.25\left(\frac{C}{D}\right)^2\right)t^*}{l}\right)\sin\left(\frac{n\pi x^*}{l}\right) \qquad \textbf{(4.19)}$$

Where l is the domain length and x direction. l=1 in the dimensionless problem.

4.3.1 Governing Equation and Boundary Conditions

The transient convection-diffusion equation is written as

$$\rho c\left(\frac{\partial T}{\partial t}+u.\nabla.\nabla T = f\right) \qquad \textbf{(4.20)}$$

The transient diffusion equation is a special case of equation (4.20) in which the convection term is zero. Where T is independent variable, u is the velocity vector, k is diffusivity, ρ is density, c is heat capacity and f is a source term. ∇denotes the gradient operator. Using the following dimensionless parameters

$$\begin{cases} T = T_0 + T^*(T_1 - T_0) \\ x^* = \frac{x}{h}, y^* = \frac{y}{h} \\ t^* = \frac{t}{t_0} \end{cases} \qquad \textbf{(4.21)}$$

where T_0 and T_1 are reference values for the field variables (e.g. temperature), t_0,is a characteristic time interval and h is a characteristic length (e.g. width of the domain) we have

$$\frac{\partial T^*}{\partial t^*} + C\nabla T^* - D\nabla.\nabla T^* = f^* \tag{4.22}$$

in which C and D are dimensionless convection and diffusion coefficients, respectively

$$\begin{cases} C = \dfrac{ut_0}{h} \\ D = \dfrac{kt_0}{h^2 \rho c} \\ f^* = f\dfrac{t_0}{\rho c(T_1 - T_0)} \end{cases} \tag{4.23}$$

Using a similar finite element discretisation for both temporal and spatial variables a two dimensional governing equation in a system of $f(t^*, x^*)$ based on equation (4.22) is considered.

$$\frac{\partial T^*}{\partial t^*} + C\frac{\partial T^*}{\partial x^*} - D\frac{\partial^2 T^*}{\partial X^{*2}} = f^* \tag{4.24}$$

Corresponding dimensionless boundary conditions for a rectangular domain are

$$\begin{cases} T^* = 0 & x^* = 0, & 0 \le t^* \ge 1 \\ T^* = 1 & x^* = 1, & 0 \le t^* \ge 1 \\ T^* = 0 & t^* = 0, & 0 \le x^* \ge 1 \end{cases} \tag{4.25}$$

4.3.2 Standard Galerkin Finite Element Scheme

The prime unknowns in the governing equations are replaced by approximate forms defined within the finite elements of the computational mesh combined with the discretisation of the problem domain. In the weighted residual finite element scheme, used in the present work, unknowns are replaced by trial function representations, which in the context of the discretised domain are given by lower order interpolation polynomials N_j (Zienkiewicz and Taylor, 1994), as

$$T \approx \widetilde{T} = \sum_{j=1}^{n} N_j T_j^* \tag{4.26}$$

Where n is total number of nodes in an element and, T_j^* is the nodal values of the unknowns. Substitution of approximate values for the unknown from equation (4.26) into the governing equation (4.24), leads to the construction of residual statements. These statements are then multiplied by appropriate weight functions (w_i) and integrated over element domains. In the standard Galerkin method, used herein, the selected weight functions are identical to the interpolation functions ($W_i = N_i$). Following the described steps we obtain

$$\int_{\Omega_e} \left[W_i \left(\frac{\partial \sum_{j=1}^{n} N_j T_j^*}{\partial t^*} + C \frac{\partial \sum_{j=1}^{n} N_j T_j^*}{\partial x^*} - D \frac{\partial^2 \sum_{j=1}^{n} N_j T_j^*}{\partial t^{*2}} \right) \right] dx^* dt^* = 0 \tag{4.27}$$

The second order differentials in equation (4.27) are reduced by the application of Green's theorem (i.e. generalised form of integration by parts). This leads to the appearance of boundary integral (flux) terms along the exterior boundaries of finite elements. For each interpolation function an identical weight function can be used to generate weighed residual equations such as equation (4.27). Therefore corresponding to a total of n interpolation functions, n equations are generated and a system of nxn equations is constructed. Using matrix notation this system is written as (Nassehi, 2002)

$$[A_{ij}]\{T_j^*\} = \{B_j\} \tag{4.28}$$

where

$$A_{ij} = \int_{\Omega_e} \left[W_i \left(\frac{\partial N_j}{\partial t^*} + D \frac{\partial N_j}{\partial x^*} \right) + C \frac{\partial W_i}{\partial x^*} \frac{\partial N_j}{\partial x^*} \right] dx^* dt^*$$

$$B_j = \int_{\Omega_e} W_i f dx^* dt^* + \int_{\Gamma_e} W_i \frac{T^{*e}}{x^*} n d\Gamma_e$$

A system of weighted residual equations should be derived for each element in the domain. This is obviously not convenient. However, by using an elemental coordinate system rather the global coordinates the uniformity of the matrix equation (4.28) can be preserved. This is achieved using isoperimetric mapping of elements of the global mesh into a master element where all the calculations are carried out (Zienkiewicz and Taylow,1994). In addition, a natural coordinate system (such as $-1 \leq \xi, \tau \leq +1$ for quadrilateral elements) can be used within the master element to enable the evaluation of all integrals within its domain by a convenient Gauss quadrature method (Gerald and Wheatley, 1984).

Repeated application of the above procedure to each element in the computations mesh leads to the construction of elemental weighted residual equations. Subsequently assembly of these equations over the common nodes between elements provides a system of global algebraic equations. The flux terms along all interior boundaries cancel each other out leaving only the boundary integrals along the exterior boundaries of the solution domain. These terms should then be treated via the imposition of boundary conditions to obtain a determinate set if equations. The solution of this set provides the model results.

4.3.3 The theta (θ) time stepping method

In this method the time derivative is kept unchanged while the weighted residual statement is used for the spatial discretisation. Therefore after discretisation, instead of a set of algebraic equations, a system of ordinary differential equations in terms of time derivatives are generated.

For a class of single step $\theta\,(0 \leq \theta \leq 1)$ methods this system can be written in matrix for as (Nassehi, 2002)

$$[M]_\theta \{T^*\}_\theta + [A]_\theta \{T\}_\theta = \{B\}_\theta \qquad \textbf{(4.29)}$$

where the subscript θ indicates that the weighted residual statement is derived at time level θ and M is mass matrix. If time derivative is written as

$$\{T^*\} = \frac{\{T^*\}_{n+\nabla t} - \{T^*\}_n}{\theta\nabla t} = \frac{\{T^*\}_n - \{T^*\}_n}{\nabla t} \qquad \textbf{(4.30)}$$

using above equations and after some algebraic manipulation we (Nassehi, 2002).

$$([M]_\theta + \theta\Delta t[A]_{n+1})\{T^*\}_{n+1} = ([M]_\theta + (1-\theta)\Delta t[A]_n)\{T^*\}_n + ((1-\theta)\{B\}_n + \theta\{B\}_{n+1})\Delta t$$

(4.31)

The results obtained by the method are in the later section with the results generated by the transient bubble function approach.

4.3.4 Elimination of boundary integrals

When bubble functions are applied the inter-element boundary integrals are not automatically eliminated during the assembly of elemental equations. This problem does not become apparent in the one dimensional case as the boundary integrals are reduced to simple nodal flux terms. The variational formulation for the transient convection-diffusion equation after the application of the Green's theorem is written as

$$\left(\frac{\partial T_n}{\partial t},v\right)+\left(C\nabla T_h,v\right)+\left(D\nabla T_h,\nabla v\right)=\left(f,v\right) \tag{4.32}$$

substitution from equation (4.32) gives

$$\left(\frac{\partial T_n}{\partial t},v\right)+\left(C\nabla T_h,v\right)+\left(D\nabla T_1,\nabla v\right)+\left(D\nabla T_b,\nabla v\right)=\left(f,v\right). \tag{4.33}$$

If v is a linear test function (weight function) according to Green's theorem (Franca and Farhat, 1995) we have

$$(\nabla v,\nabla \phi)_{\Omega_e}=-(\Delta v,\phi)_{\Omega_e}+(\nabla v,\phi)_{\Gamma_e}=0 \tag{4.34}$$

where is a bubble function. Therefore equation (4.33) is reduced to

$$\left(\frac{\partial T_n}{\partial t},v\right)+\left(C\nabla T_h,v\right)+\left(D\nabla T_1,\nabla v\right)=\left(f,v\right)$$

as can be seen the bubble function does not affect the Laplacian term in the convection-diffusion equation and therefore no boundary integral due to the bubble function exists.

4.3.5 Results

The main objective of the present work has been the construction of the novel scheme for the transient transport problem to allow temporal and spatial discreitsation. In addition it is shown that fundamentally, it is possible to use bubble functions to have stable time approximations under unseen conditions. Analytical solutions are obtained for a bench mark problem to validate the numerical results. This comparison shows the ability of the past scheme to generate theoretically expected simulations. Similarly, the theta time stepping method is used to evaluate the model performance against widly used classical techniques on rthe realistic conditions. In all figures b and b_t are bubble coefficient and l_x and l_t indicate the element length in x and t directions, respectively.

Figure 4.17 shows the Domain and its boundary. Figure 4.18 shows the finite element mesh.

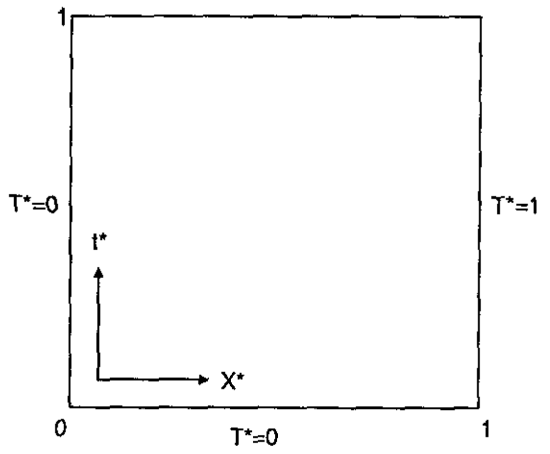


Figure 4.17 Domain and its boundaries

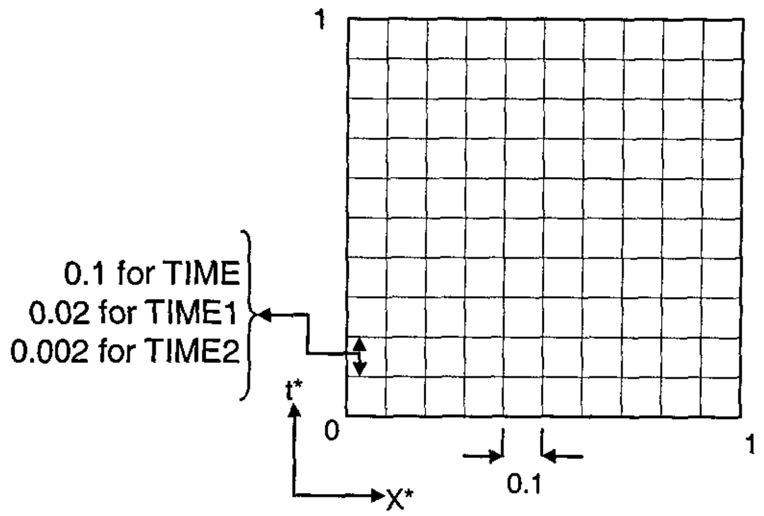


Figure 4.18 Finite element mesh schemes

As temporal and spatial discretisation is used for t and x discretisation and the results are computed, while three different t mesh schemes is used.

Figures 4.19 to 4.25 show the results for the diffusion problem. For D=1 and l_t = 0.1 and 0.02 the transient solution is unstable while with l_t = 0.002 the accurate-stable solution can be achieved. As figure 4.19 shows the instability for l_t = 0.002 is minor.

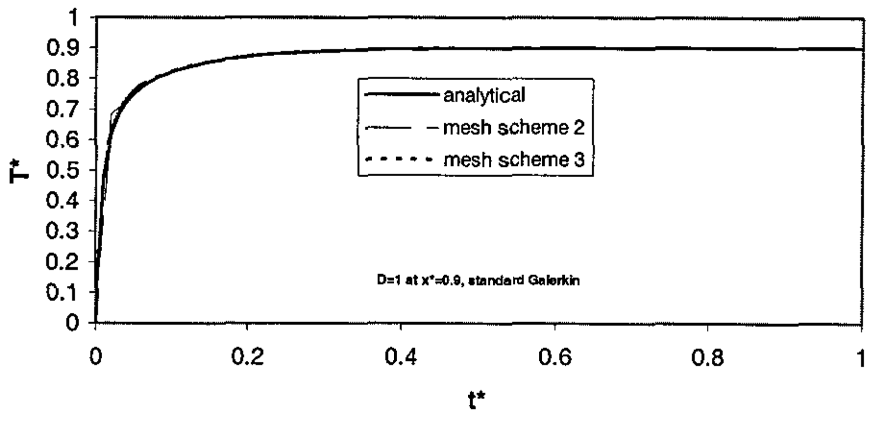


Figure 4.19 Transient response of diffusion equation at D=1 and x*=0.9. Mesh schemes 2 and 3.

The instability in figure 4.20 demonstrates the multi-scale behaviour which is however, stabilised using the bubble enriched element.

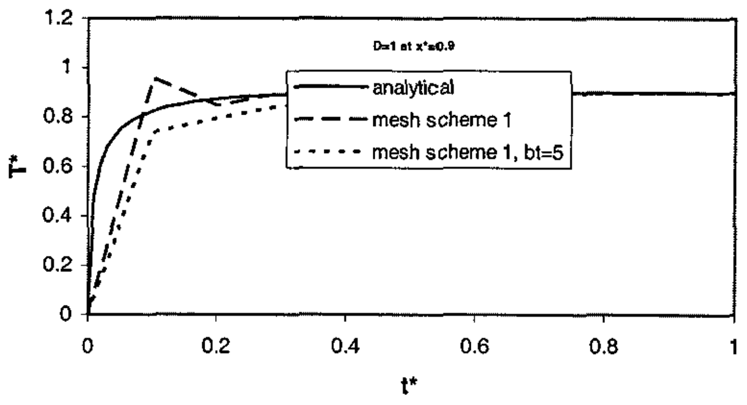


Figure 4.20 Transient response of diffusion equation at D=1 and x*=0.9 with and without temporal bubble function.

Figure 4.21 shows at D = 5 the multi-scale behaviour increases while with $l_t = 0.002$ the solution is accurate and stable. To stabilise the solution for $l_t = 0.1$ and 0.02 bubble functions are applied.

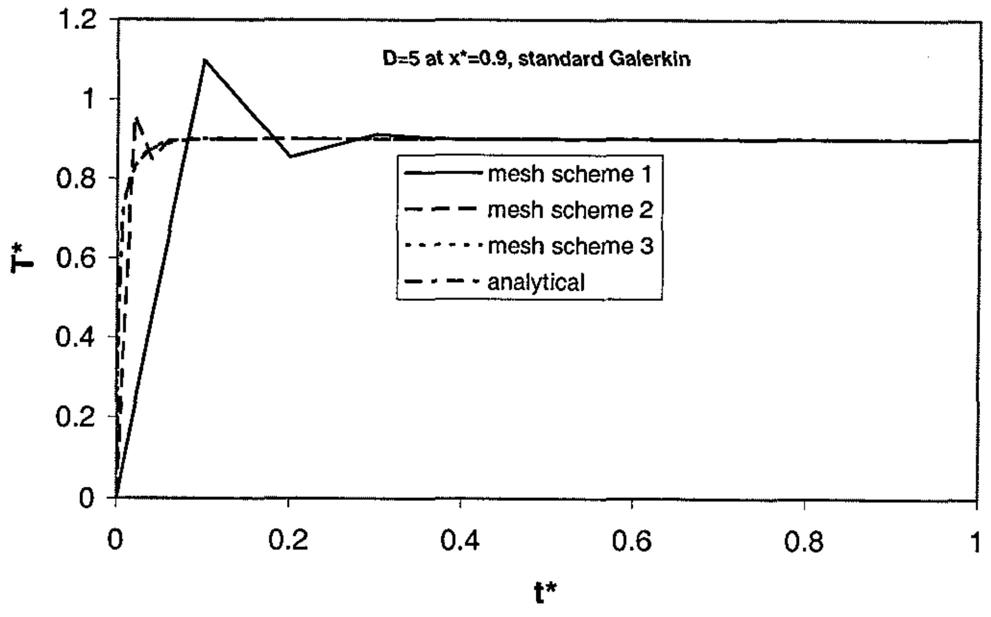


Figure 4.21 Transient response of diffusion equation at D=5 and x^*=0.9 for all mesh schemes.

Figures 4.22 and 4.23 show the results of two types of bubble functions based on $\phi_\tau = \sum_{q=1}^{n} \left(1-\tau^2\right)^q$ and $\phi_\tau = \sum_{q=1}^{n} \left(1-\tau^{2q}\right)$, respectively.

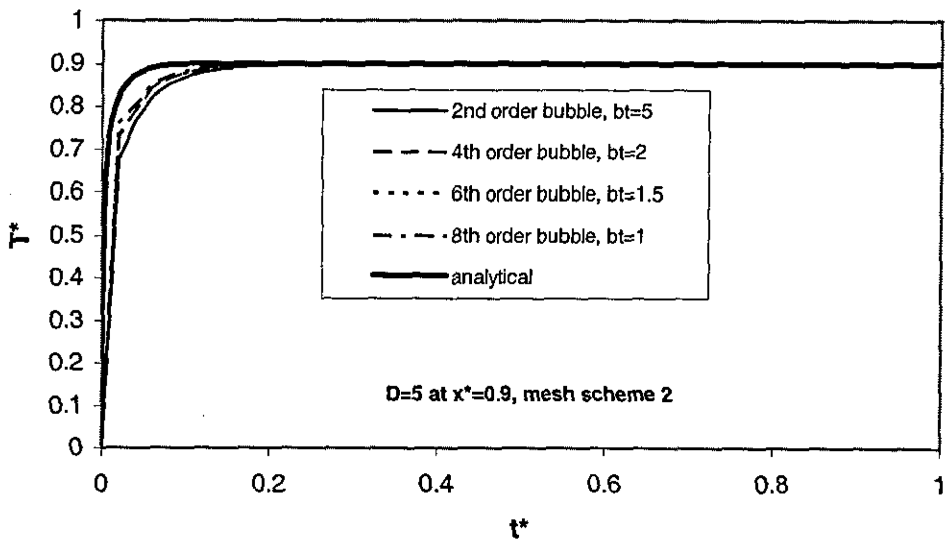


Figure 4.22 Transient response if diffusion equation at D=5 and x^*=0.9 using different orders of the temporal bubble function.

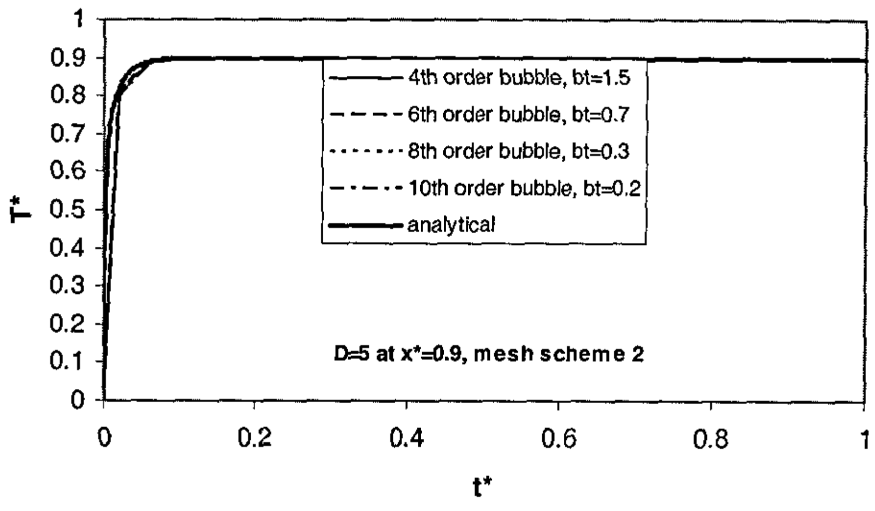


Figure 4.23 Transient response of diffusion equation at D=5 and x^*=0.9 using different orders of the temporal bubble functions.

Figure 4.23 indicates that the bubble of the form in the Equation (3.119) perform better in all simulations the 4th order bubble of this type is used. With the same time steps the theta method at $\Delta t^* = 0.002$ gives the accurate solution (figure 4.24). By increasing diffusion coefficient to D=10 the multi-scale behaviour increases and to achieve stable solution bubble functions are applied.

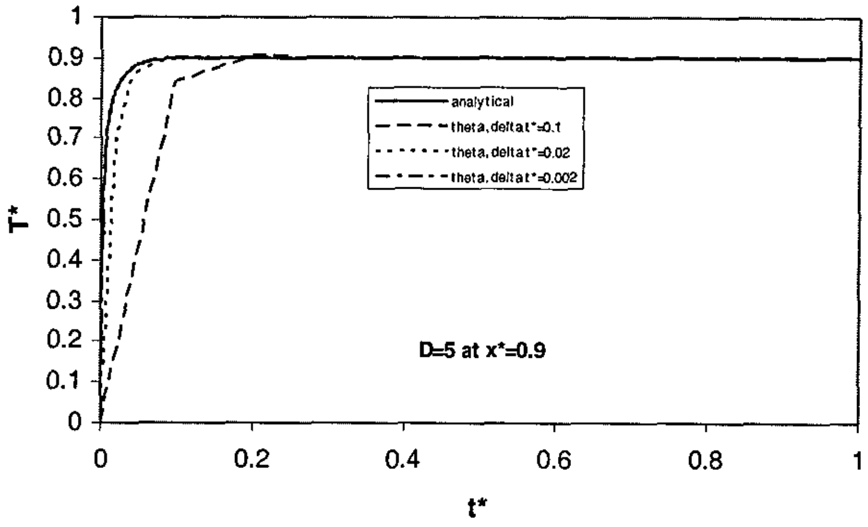


Figure 4.24 Transient response of diffusion equation using theta method at D=5 and x^*=0.9

For mesh scheme 2 (l_t = 0.02), as figure 4.25 shows the stable solution can be achieved at b=2. it must be noted that while the problem has no spatial multi-scale behaviour, the temporal behaviour is quite different. While diffusion is the only transport phenomena in x direction, and therefore no multi-scale behaviour in each direction is quite different and it is why the bubble function for time is added to the bubble function for spatial direction. for multi-dimensional steady problem and elemental bubble function is used (Parvazania et al.,

2006) as problem shows a unique behaviour in all dimensions. For transient problems the dynamic behaviour in time is different and for time, as Equation (3.119) shows, the bubble function is added to the bubble for spatial dimension.

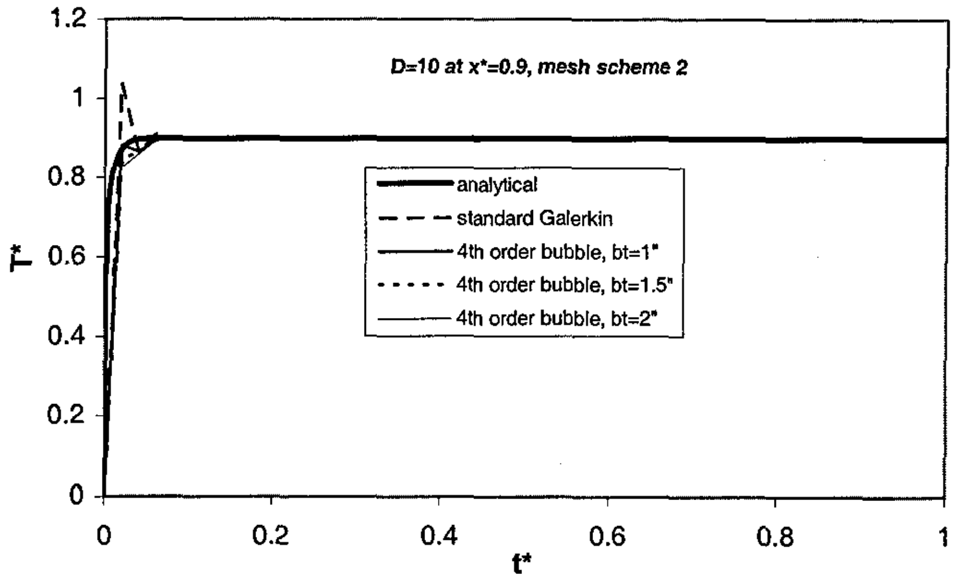


Figure 4.25 Transient response of diffusion equation at D=10 and x^*=0.9. Mesh scheme 2 with and without temporal bubble function.

Figures 4.26 to 4.35 show the results for the convection-diffusion equation. In these cases in both temporal and spatial directions the problem may show multi-scale behaviour. Multi-scale finite element modelling of convection-diffusion problem has been discussed previously by Parvazinia et al. (2006b) in detail. At C=5 using the mesh scheme 3 (l_t=0.002) the stable solution can be achieved (figure 4.26) and the solution is similar to the theta method with Δt^*=0.002 (figure 4.27).

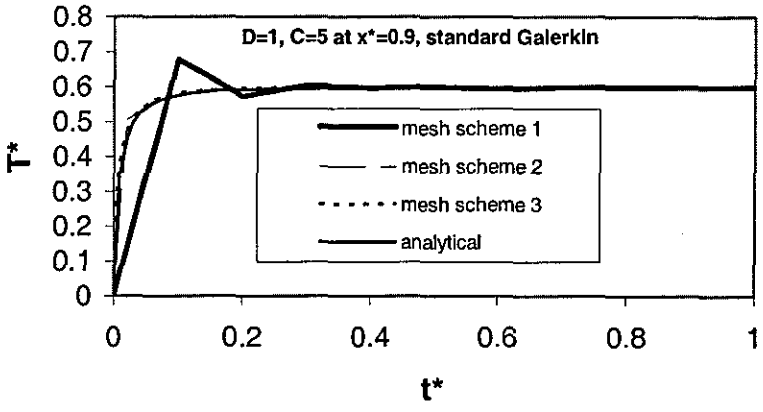


Figure 4.26 Transient response of convection-diffusion equation at at D=1, C=5 and $x^*=0.9$ for all mesh schemes.

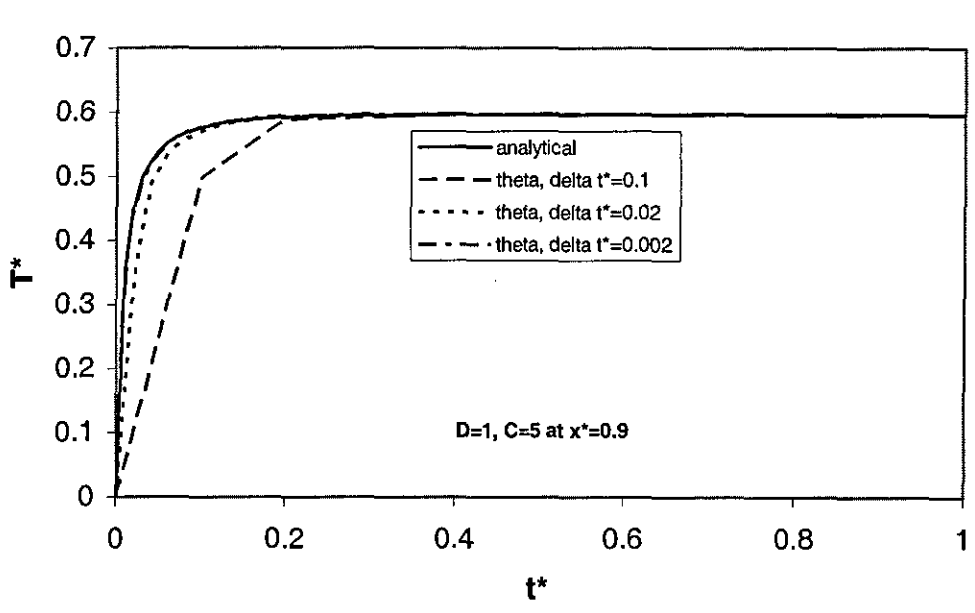


Figure 4.27 Transient response of convection-diffusion equation theta method at D=1, C=5 and $x^*=0.9$

At C=10, as figure 4.28 shows, mesh scheme 1 and 2 can yield stable solution only with bubble function while the mesh scheme 3 serves the stable-accurate results. It shows that the temporal discretisation for mesh scheme 3 is fine enough to over come the multi-scale behaviour.

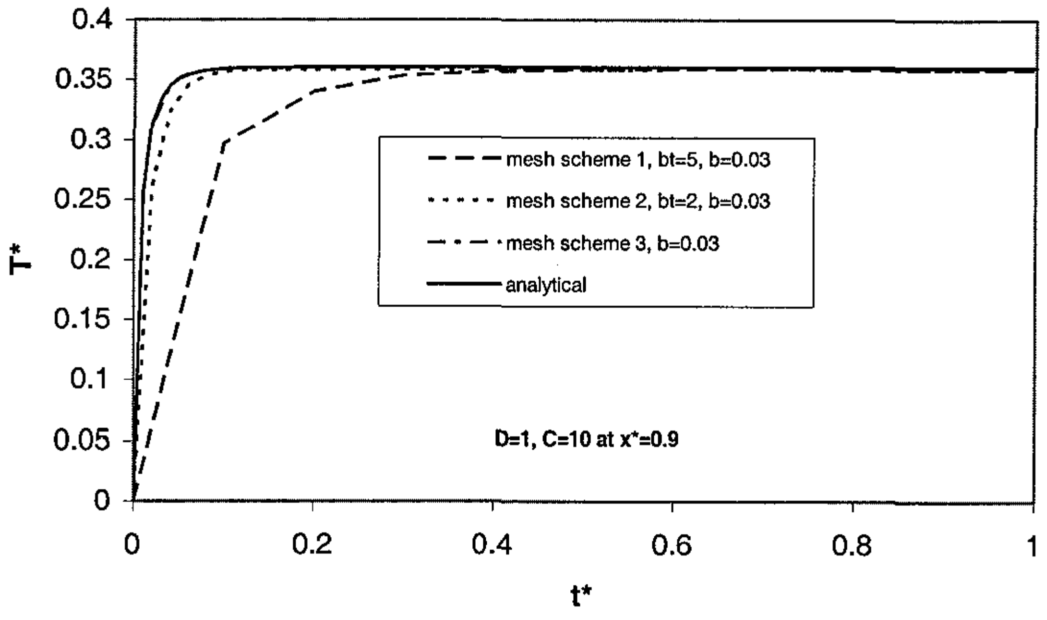


Figure 4.28 Transient response of convection-diffusion equation using theta method at D=1, C=10 and x^*=0.9. Mesh schemes 1 and 2 with temporal and spatial bubble function and mesh scheme 3 with just spatial bubble function.

Figure 4.29 shows the results for theta method. Although the stable solution can be achieved by theta method but at C=10 the solution is slightly under estimated according to the exact solution.

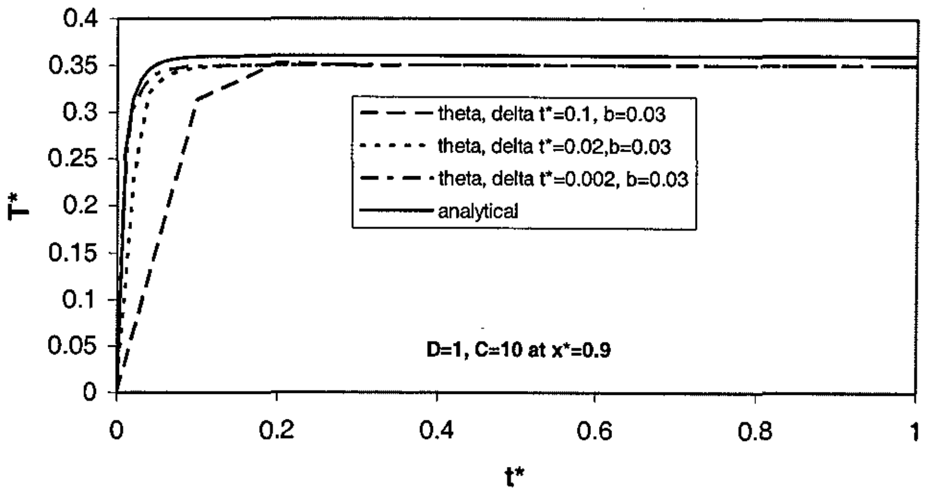


Figure 4.29 Transient response of convection-diffusion equation using theta method at D=1, C=10 and x^*=0.9 with spatial bubble function.

At C=50, as figure 4.30 shows even with the mesh scheme 3 the solution is slightly unstable. Using bubble function the stable solution can be achieved as figures 4.30 and 4.31 indicate.

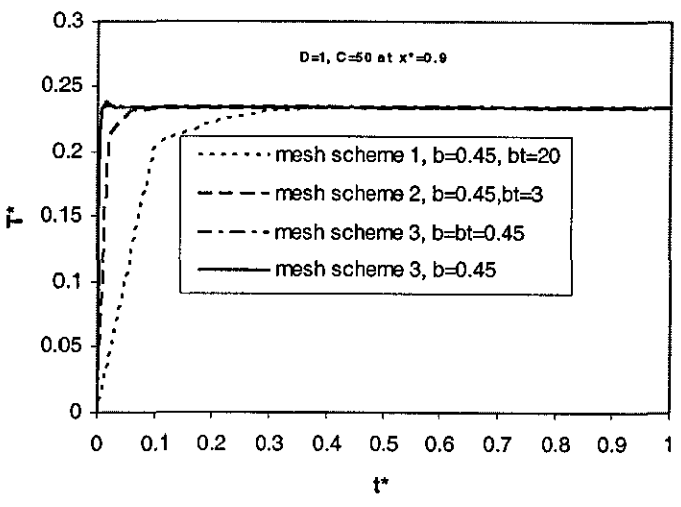


Figure 4.30 Transient response of convection-diffusion equation using theta method at D=1, C=50 and x^*=0.9 for all mesh schemes with spatial and temporal meshes.

As figure 4.31 shows with theta method the solution is slightly under estimated. Therefore with respect to results for theta method at C=10 and 50 it seems although the solution is stable but it slightly under estimates when the convection coefficient is more than 10.

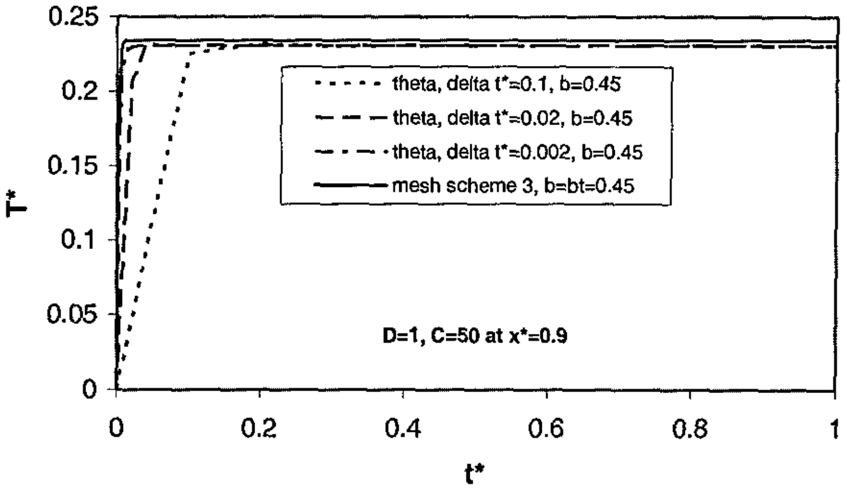


Figure 4.31 Comparison of transient response of convection-diffusion equation using theta method at D=1, C=50 and x^*=0.9 with spatial bubble function and mesh scheme 3 with both spatial and temporal bubble functions.

It must be noted that since at C=50 the exact solution is nearly zero an over-diffusive stable is intentionally used to show the results in *t* direction (see figure 4.32).

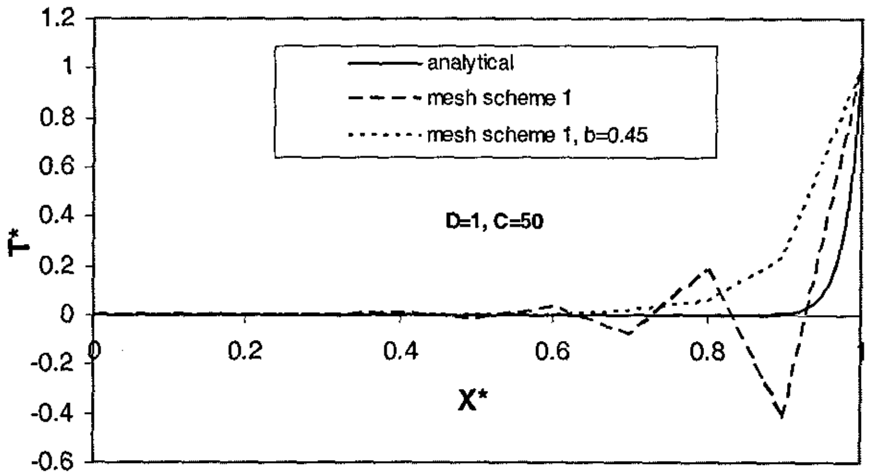


Figure 4.32 Steady solution of the convection-diffusion equation at D=1, C=50 and x^*=0.9 using mesh scheme 1 with and without spatial bubble function.

As figures 4.33 and 4.34 show the temporal and spatial multi-scale behaviour are different.

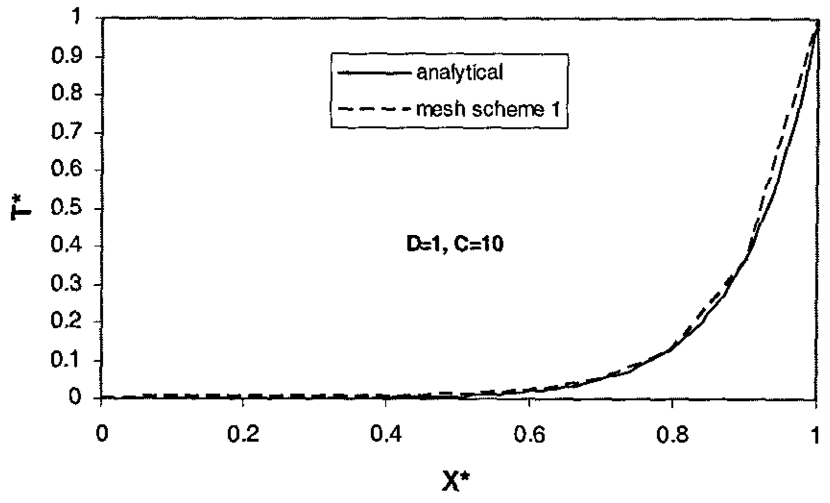


Figure 4.33 Steady solution of the convection-diffusion equation at D=1, C=10 and x^*=0.9. No multi-scale behaviour in x direction is observed.

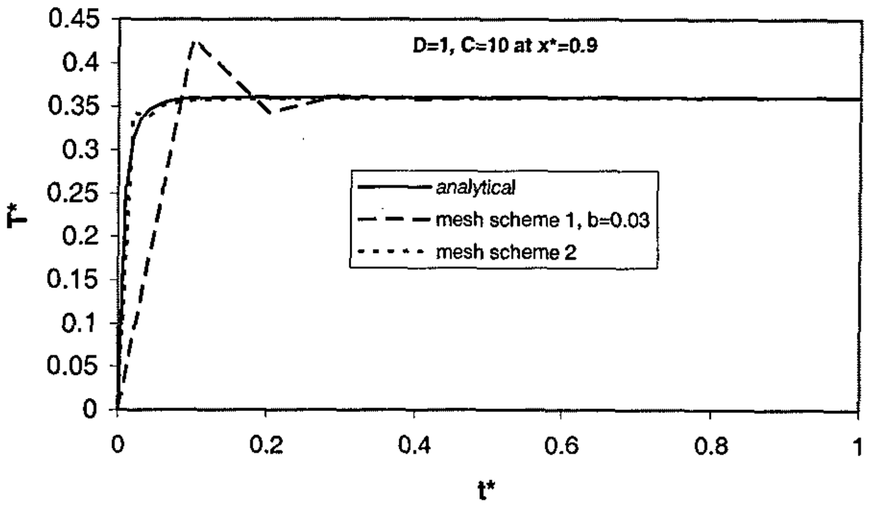


Figure 4.34 Transient response of the convection-diffusion equation at D=1, C=10 and x^*=0.9 with strong temporal multi-scale behaviour.

While in the x direction the solution is stable and very close to the exact solution at C=10 and l_x=0.1 (figure 4.33) the solution is quite unstable in time with the same discretisation (figure 4.34 curve mesh scheme 1). As figure 4.34 shows even with l_x=0.02 using mesh scheme 2 the solution is still unstable. These results confirm that in time the behaviour is extremely multi-scale.

Comparing figures 4.28 and 4.30 shows that by increasing the convection coefficient from C=10 to C=50, stable solutions can be achieved by increasing the bubble coefficient. Therefore, if we look at the bubble coefficient as a measure of multi-scale behaviour it can be found that at C=50 the stable solution in x direction can be obtained by b=0.45 (figure 4.32) while for t with the same level of discretisation bubble coefficient b_t=20 (figure 4.31). It shows clearly the different dynamics and so different level of multi-scale behaviour in x and t directions.

The reason for the highly multi-scale behaviour in time can be found in the analytical solution. For diffusion problem while no spatial multi-scale behaviour exists, the transient response is determined by $\exp\left(\frac{n^2\pi^2Dt^*}{l}\right)$. The exponential argument becomes large in small amounts of D and it is the source of the strong multi-scale behaviour and instability in transient response. For convection-diffusion problem same situation is observed. The spatial dynamics behaviour is affected by $\exp\left(0.5\frac{C}{D}\left(x^*-l\right)\right)$ and when C becomes large the multi-scale behaviour is observed. The temporal dynamics behaviour is affected by $\exp\left(\frac{-\left(n^2\pi^2+0.25\left(\frac{C}{D}\right)^2\right)t^*}{l}\right)$ in which $(C/D)^2$ becomes large at small values of (C/D) and therefore, strong temporal multi-scale behaviour is observed.

On the other hand for both equations when l becomes small the argument becomes larger. In finite element discretisation $l=l_x$ and with finer refinements in x direction the solution sows temporal instability (stronger multi-scale behaviour) as shown in figure 4.36. Figure 4.36 shows using mesh scheme 2 (l_x=0.1, l_t=0.02) the solution with b_t=2.5 is unstable while for mesh scheme 4 (l_x=0.2, l_t=0.02) (figure 4.35) the solution, with the same bubble coefficient for time b_t, is stable.

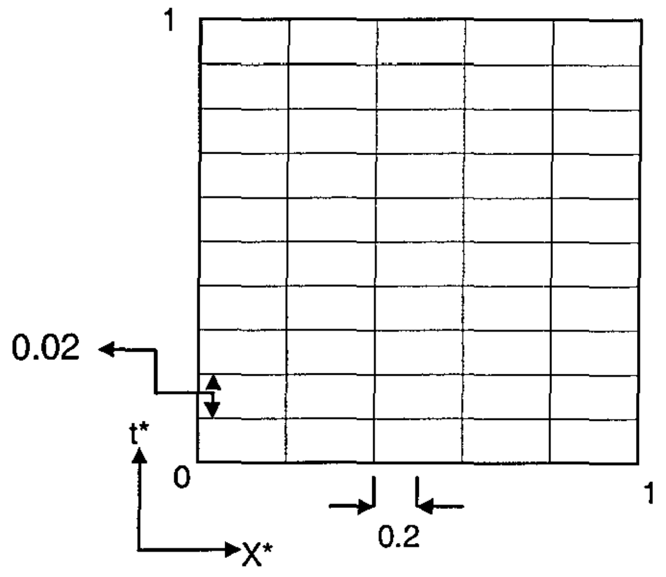


Figure 4.35 Mesh scheme 4

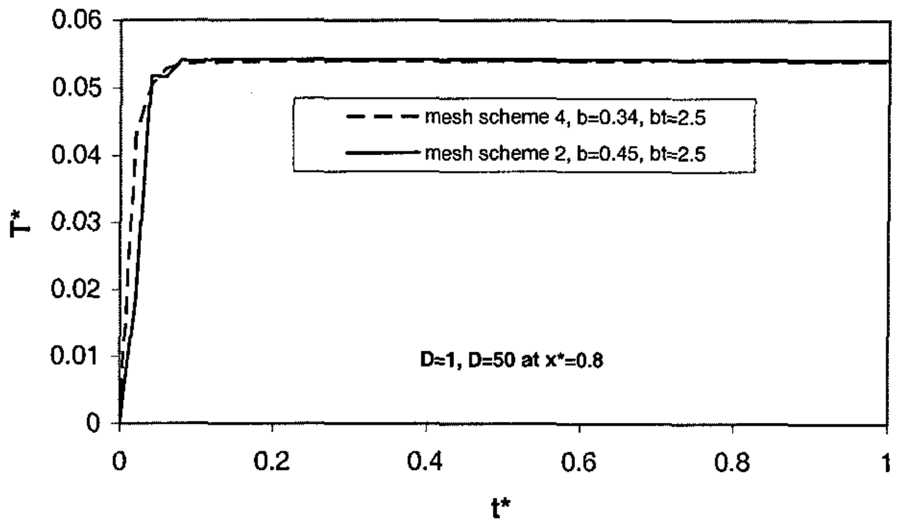


Figure 4.36 Transient response of the convection-diffusion equation at D=1, C=50 and x^*=0.8 using different x refinement of mesh schemes 2 and 4 with both spatial and temporal bubble functions.

In usual time stepping methods for transient convection-dispersion equation it has been shown that the stability of standard Galerkin finite element method is conditional (Nassehi, 1981). The stability can be derived by $\frac{\Delta x^*}{\Delta t^*}$ relating to the eigen values of the stiffness matrix which shows stability is affected by the element lengths.

4.3.6 Conclusion

A series of numerical experiments supported by the analytical solution are used to evaluate the performance of a novel scheme. The results are compared with those obtained by the classical theta time stepping technique. The proposed method is capable of yielding stable results for transient diffusion and transient convection-diffusion equations. In comparison with theta method although this method gives stable solution in the theta time stepping method (and similar time stepping methods) by increasing the transport coefficients (C or D) the time intervals must be decreased and computational cost increases. In the proposed method since the scheme solves the problem as a boundary value problem it is very cost effective. In addition the method has all the advantageous of the finite element discretistation for the time approximation.

4.4 Bench Mark Problem 3

Milford Haven estuary, located in West Wales (U.K.), in its upper parts comprises of a network of relatively narrow channels connecting the inland waters of Eastern and Western Cleddeau and several smaller rivers to the sea (figure 4.20). Altogether branches of this waterway enclose 110 km of varied coastline. The estuary channel is 2.5 km wide at its mouth.

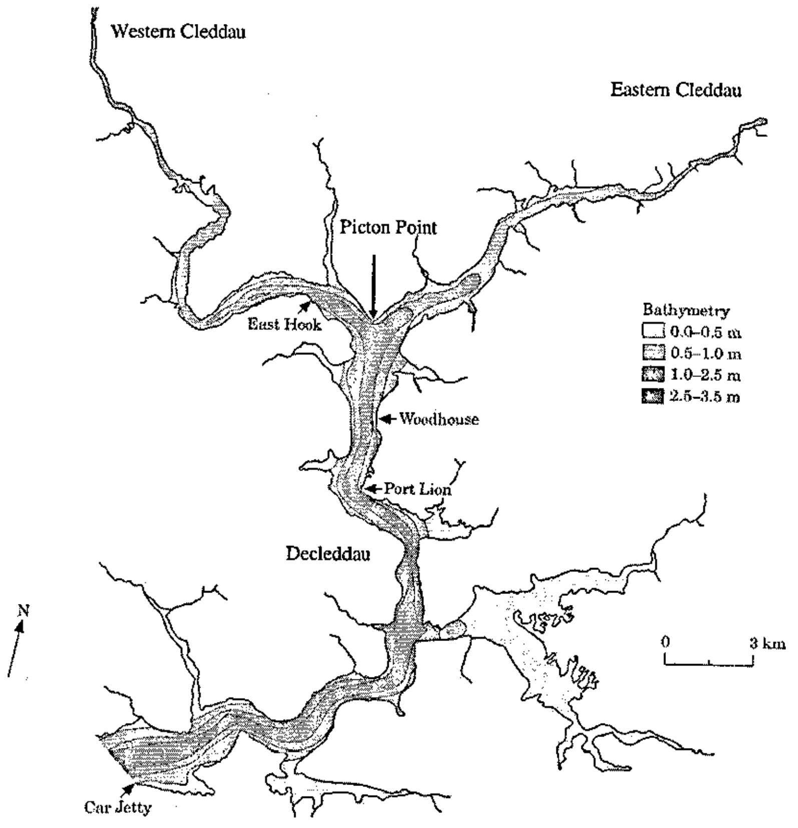


Figure 4.37 Map of Milford-Haven Estuary

Beyond Carr Jetty and above the road bridge between Pembroke Dock and Neyland the narrowing main channel of the estuary (from this point landwards called Daucleddau) turns northwards until reaching a limit at the junction of Eastern and Western Cleddau rivers at Picton Point, about 27 km from the estuary mouth. Along the described upper part of the estuary water flow is predominantly one-dimensional. However, significant hydro-environmental analysis are not uncommon with in this upper section of the Milford-Haven estuary. Many tributaries flow into Daucleddau and Easter and Western Cleddau rivers. These usually form inlets which initially are of a similar width to the main channel, but are much shallower and generally their bed become almost completely exposed during low water (West, 1978). Therefore these shallow inlets are thought to have negligible effect on the longitudinal momentum transfer.

The limit of tide propagation for spring tides in the Western Cleddau river is about 11 km (in the region of Crowhill Weir with a range of 3.9 above ordnance datum A.O.D.) and about 8 km in the Eastern Cleddau river (in the region of Canaston Weir a range of 4.2 m A.O.D.) from Picton Point, respectively. The mean tidal ranges at Milford Haven are 6.3 m and 2.7 m for spring and neaps, respectively.

For both spring and neap tides a phase lag of maximum 36 min. in the occurrence of high and low water between the port of Milford Haven and locations near the tidal limits of the Western and Eastern Cleddau rivers is recorded (Nelson-Smith, 1965)

The bed level of the Daucleddau rise from 15 m below ordnance datum (B.O.D.) near Lawrenny to about 5 m B.O.D. near Picton Point. The average width of the main channel is of the order of 500 m. However, the Eastern and Western Cleddau rivers are regarded as very narrow as their width gradually decrease to a few meters towards their tidal limits. There is very little water in these rivers during low water springs except in a few isolated pools.

Description of sediment distribution in the upper Milford-Haven is reported by Williams (1971) and West (1979). In the Eastern and Western rivers flow channel bed is mainly covered by mud in the Daucleddau region the bottom surface consists of sand and rock.

Fresh water discharges of Cleddau rivers is usually insignificant in comparison with the tidal flows into this estuary which is of the order of 10^4 m^3 s^{-1} at Hobbs Point at mid-tide (Gunn and Yenigun, 1985) for example, the recorded discharges during the 1977 survey of the upper estuary (West, 1978) the fresh water discharged of the Cleddau rivers were of the order of 1 m^3 s^{-1}. Another survey in spring of 1979 (Williams and Nassehi 1980b) gives the mean daily fresh water flows of Western and Eastern Cleddau river as 6 m^3 s^{-1} and 5.5 m^3 s^{-1}, respectively.

As the Eastern and Western Cleddau rivers are roughly of the same size therefore the Upper Milford-Haven Estuary should be regarded as a branched water way. Therefore the governing equations used to describe hydrodynamic conditions in this water system should be supplemented by mass and momentum balance relationships representative of river junctions. Many one and two-dimensional models for the simulation of tidal wave propagation in the Upper Milford-Haven models can be found in the literature. These models use the explicit Leap-Frog and the four point implicit finite difference schemes as well the finite element schemes (e.g. see Nassehi and Williams, 1986, Nassehi Bikangaga, 1993, Das and Nassehi, 2004). After conducting the usual procedures for the calibration and verification of hydrodynamic models these models yield outputs regarding the flow depth and water velocity. The values of Manning's coefficient obtained after calibration of one dimensional are somewhat different. For example, using a one dimensional approach based on the four point implicit scheme Nassehi and Williams (1986) give the values of this coefficient to be 0.018 for the DeCleddau region rising to 0.020 for the Eastern and Western Cleddau rivers. However, the most suitable set of Manning's coefficient for a one dimensional Taylor-Galerkin scheme has been found by Nassehi and Bikangaga (1993) to vary from 0.017 at the downstream end of the DeCleddau to 0.026 for the Eastern and Western Cleddau rivers. Such discrepancy points to the approximate nature of one dimensional modelling which as mentioned earlier is only suitable for obtaining fast estimates for hydrodynamic phenomena. It should never the less, be stressed that such data provides very valuable information to formulate useful management policies for complex problems encountered in natural water systems. Although detailed and very accurate quantitative data obtained via sophisticated models are always useful they may not be absolutely necessary for making sound management decisions. Typical calibration and verification results for Port lion and East Hook stations are shown in figures 4.21a and 4.22b, respectively. The calibration event was spring tide of 24th April, 1979 and for the verification the water surface elevations recorded during the neap tide of 2nd May, 1979 are used.

As mentioned earlier construction, calibration and validation of hydrodynamic models for natural water systems is often the precursor of the development of pollutant dispersion models. The velocity field obtain from hydrodynamic models provides one of the essential data that should be inserted as an input to a transport model. In the following section the development of a solution scheme dispersion model for the solution the Upper Milford-Haven Estuary is discussed.

Different methods used in order to obtain solute transport of Upper Milford-Haven Estuary have provided results with various degrees of accuracy see table.

4.4.1 Governing Equation

A two dimensional solute transport model involves solution of depth averaged convection-dispersion equation for determining distribution of a solute in the flow domain. In a horizontal Eulerian system, the solute transport is expressed as

$$\frac{\partial(ch)}{\partial t} + \nabla.(ch\underline{u}) = \nabla.(hD\nabla.(c)) \qquad \textbf{(4.35)}$$

Where D is the dispersion coefficient (ML^{-2}) and h is the total water depth. For a complex tidal water system such as an estuary, the appropriate values of dispersion equation in equation (4.35) should be selected using realistic data. The task typically involves the use of an empirical approach that utilises available field survey data (e.g. West and Boyd, 1981). For reactive pollutants an appropriate reaction term should also be added to the equation (4.35).

4.4.2 Solution using Taylor-Galerkin Finite Element scheme

Numerical solution of a convection dominated equation such as the solute transport equation in a stationary (Eulerian) framework is in general prone to yield unstable results. For example, it is well know that the standard Galerkin finite element solution of equation (4.35) in a fixed co-ordinate system only yields stable results for cases where the convection is small. Therefore in convection dominated situations, such as prevailing conditions in a flow region where dispersion tends to be very small, some form of numerical dissipation (upwinding) must be used. Numerical dissipative Petrov-Galerkin schemes utilising special weight functions are commonly used in Eulerian frameworks to obtain stable simulations for convection dominated transport problems. Consistent streamline upwind Petrov-Galerkin (SUPG) scheme has been show to generate accurate results in many flow problems. However, because in multidimensional domains it is not possible to evaluate, with certainty, an optimum upwinding parameter the SUPG scheme cannot always be expected to generate accurate results. In most cases, the application of upwinding, whilst ensuring the stability of the numerical solutions, has a detrimental effect on the accuracy generating over damping results.

This problem can be avoided if a moving (i.e. Lagrangian) framework in which the convection terms naturally disappear from the governing equations is employed. In order to develop an effective scheme in a Lagrangian system the computational grid must be continuously be regenerated to avoid excessive mesh distortions. This provides a very robust computational method but at a high computation cost. In this section a Lagrange-Galerkin finite element scheme based on the utilisation of Reynolds transport theorem (Aris 1989) for the solution of equation (4.35) is presented. Similar to the hydrodynamic equations this scheme is based on a framework constructed along the fluid particle trajectories.

After the application of Reynolds transport theorem the variational statement of the convection0dispersion equation (4.35) can be written as

$$\int_{\Omega_{n+1}} N(ch)^{n+1} d\Omega_{n+1} - \int_{\Omega_n} N(ch)^n d\Omega_n = \int_{t_n}^{t_{n+1}} \int_{\Omega_t} N[\nabla.(hD.\nabla c)] d\Omega_t dt \qquad \textbf{(4.36)}$$

Where $N$is a trial function and Ω_{n+1}, Ω_n are the domains occupied by the fluid at time levels t $_{n+1}$ and t $_n$ respectively. After the application of integration by parts to the right hand side of equation (4.36) and the assumption of zero flux on the boundaries, that is setting

$$\int_{\Gamma} Nhn.D.\nabla c d\Gamma = 0 \qquad \textbf{(4.37)}$$

Where Γ is the domain boundary. We get

$$\int_{\Omega_{n+1}} N(ch)^{n+1} d\Omega_{n+1} - \int_{\Omega_n} N(ch)^n d\Omega_n = -\int_{t_n}^{t_{n+1}} \int_{\Omega_t} h\nabla N(D.\nabla c) d\Omega_t dt \qquad \textbf{(4.38)}$$

The time intergrals in equation (4.38) can be approximated using the θ method (Nassehi 2000). After this stage the usual finite element discretisation of equation (4.38) gives

$$\sum_J \left(M_{IJ}^{n+1} + \theta\Delta t_n K_{IJ}^{n+1}\right)(ch)_J^{n+1} = \sum_J \left[M_{IJ}^n - (1-\theta)t_n K_{IJ}^n\right](ch)_J^n \tag{4.39}$$

In equation (4.39) J is the row index which is also the index of the weight function used in the derivation of the weighted residual finite element stiffness equations. A summation over this index indicates assembly of the elemental stiffness equations. Index I represents the number of nodal degrees of freedom in the finite elements used for the discretisation. Using C^0 tensor product Lagrange elements a field variable can be represented, say, by bi quadratic shape functions as $\tilde{f} = \sum_{I=1}^{9} f_I N_I$, where N_I are the shape functions. In the Galerkin method the weight functions are the same as the shape functions and, hence, M_{IJ}^{n+1} and M_{IJ}^n are mass matrices at time levels t_{n+1} and t_n respectively, given as

$$M_{IJ}^{n+1} = \int_{\Omega_{n+1}} N_I N_J d\Omega_{n+1} \text{ and } M_{IJ}^n = \int_{\Omega_n} N_I N_J d\Omega_n \tag{4.40}$$

And K_{IJ}^{n+1}, K_{IJ}^n are the dispersion-stiffness matrices derived as

$$K_{IJ}^{n+1} = \int_{\Omega_{n+1}} h\nabla N_I . D^{n+1} . \underline{\nabla} N_J d\Omega_{n+1} \text{ and } K_{IJ}^n = \int_{\Omega_n} h\nabla N_I . D^n . \underline{\nabla} N_J d\Omega_n$$

Here $(ch)_J^{n+1}$ are the nodal values at the fixed mesh at time level t_{n+1} and $(ch)_J^n$ are the corresponding nodal values at the distorted mesh (constructed by shifting the original mesh points along the path-lines) at time level t_n.

In tidal water domains, where the flow regime can be effectively considered to be 'compressible', measures of integrations in equation (4.40) (i.e. areas represented by $d\Omega_{n+1}$

and $d\Omega_n$) are in general not equal. Total water depth (h) and, in some cases, the dispersion also changes with time. This means that the spatial derivatives of the shape functions used in the evaluation of the terms of the stiffness matrices corresponding to the dispersion terms should be recalculated at each time level. This problem can be resolved by the use of a fully implicit time stepping (i.e. using $\theta = 1$) scheme.

4.4.3 Results

The above scheme is used in conjunction with the bubble function. The results obtained by this method and other schemes previously published are compared below.

Table 4.1 Compares simulated and observed salinities for different schemes (numbers show the percentage of discrepancy between the computed and observed data).

scheme	Spring tide (25 April 77)				Neap tide (2 May 77)			
used	D.Cleddau & W.Cleddau		D.Cleddau & E.Cleddau		D.Cleddau & W.Cleddau		D.Cleddau & E.Cleddau	
	h.w.	l.w.	h.w.	l.w.	h.w.	l.w.	h.w.	l.w.
FD	11	4.5	9.6	8.8	5.4	27.2	2.4	4.5
TG	12.9	8.6	10	17.8	NA	NA	NA	NA
LG1	5.4	4.3	6.5	2.3	4.5	8	2.8	3.1
LG2	4.2	2.9	4.6	1.7	3.3	6.2	1.9	2.8

Table 4.1 FD: finite difference (Nassehi and Williams, 1986) TG: Taylor-Galerkin finite element scheme (Nassehi and Bikangaga, 1993); LG1: Lagrangian-Galerkin scheme with arbitrary division of junction at the confluence (Kafai and Nassehi, 1999); LG2: Lagrangian-Galerkin scheme with additional mass balance at the junction area (Das and Nassehi, 2004).

The following graphs (figures 4.37 to 4.39) show the results obtained using the bubble function method developed in this study which proves superiority of the method over the methods used.previously.

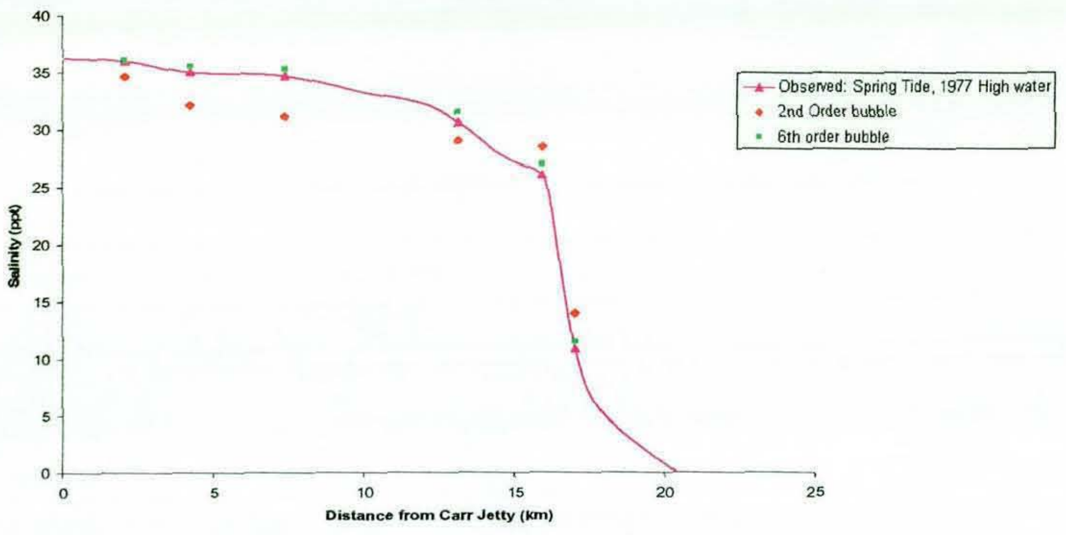


Figure 4.38 Comparison of simulated and observed longitudinal salinity at high water of Daucleddau and East Cleaddau

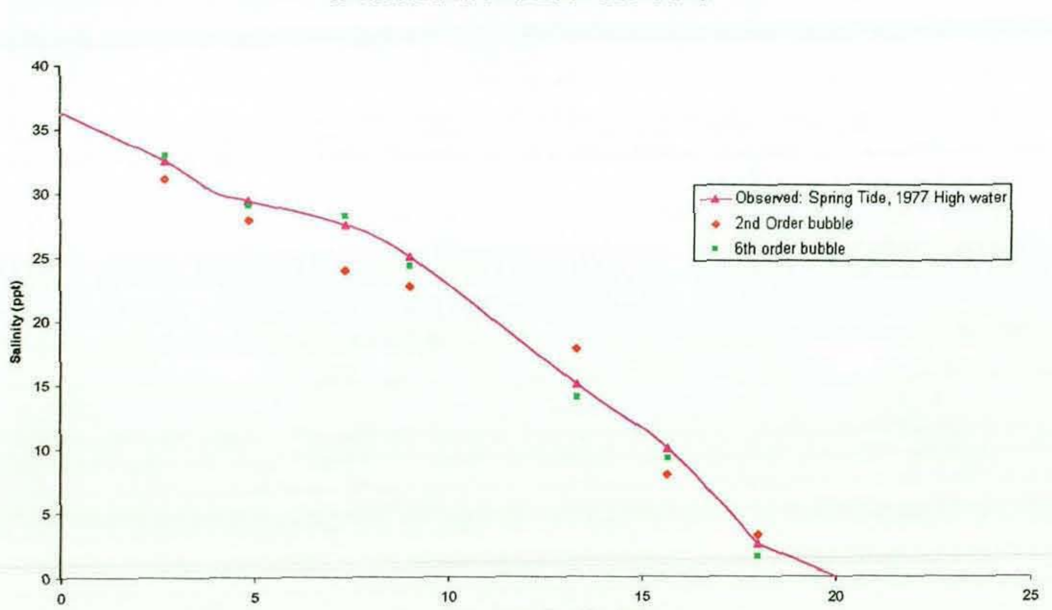


Figure 4.39 Comparison of simulated and observed longitudinal salinity at high water of Daucleddau and West Cleaddau

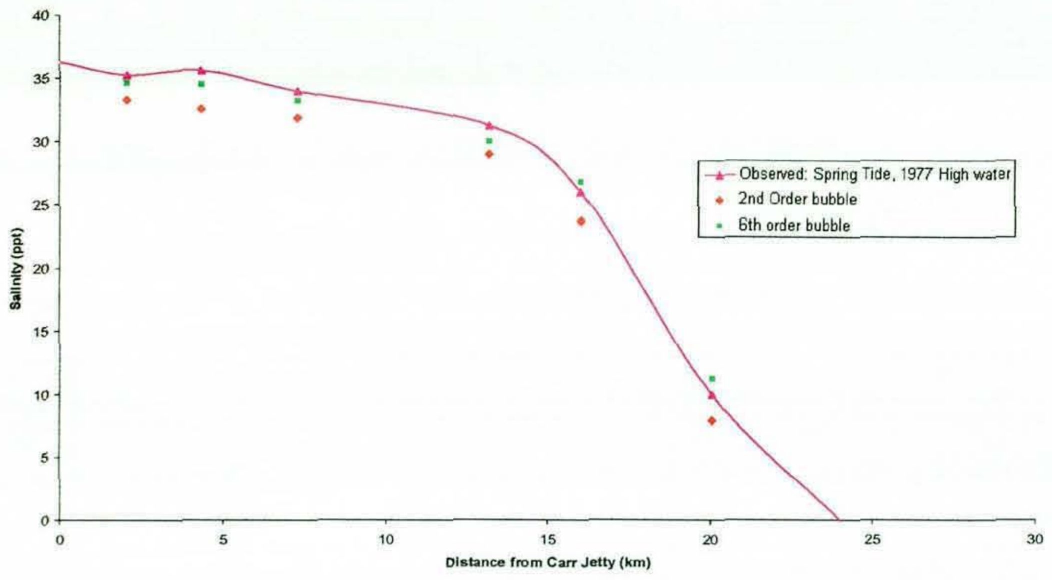


Figure 4.40 Comparison of simulated and observed longitudinal salinity at high water of Daucleddau and West Cleaddau

4.4.4 Conclusion

Figures 4.42 to 4.44 show very close comparison between the field observations and results obtained by the bubble function method provide that a sixth order is used. However even using a lower order bubble function the obtained results are at least comparable with most of the results shown in the table 4.1 which are generated using various finite difference or finite element techniques, previously published.

In this chapter results obtain by the bubble function method for a number of multi-scale problems both for steady state and transient conditions are described. The part with in this chapter where comparison of the bubble function simulation with an actual convection dominated multi-scale transport problem is presented as the most significant proof regarding the accuracy and applicability of the method developed using the past results.

Chapter 5
Conclusions and Future work

Traditional numerical schemes used for the modelling of field problems in engineering have difficulties in coping with phenomena associated with multi-scale flow behaviour. Techniques based on using finite difference, finite volume or finite element methods have relied on the use of excessive mesh refinement to overcome such difficulties. Although excessive mesh refinement may solve many multi-scale problem, however, such solutions are impractical, due to the large amount of computational time that is usually necessary in these approaches.

This research concentrated on finding a method that did not entail the use of extravagant amounts of computational power and time. The method discussed and examined for its usability to cope with the multi-scale flow phenomena was the extension of the newly emerging bubble function method.

5.1 Conclusions

The method developed in this research is using the Galerkin finite element technique in conjunction with the bubble enriched finite elements (i.e. bubble function method). This allows the selective use of more accurate interpolation functions than the ordinary finite elements. The selective use of higher order interpolation functions enhances the capability of the finite element schemes to cope with significant changes in field unknowns over small areas of the problem domain. Such behaviour is encountered in porous and convection dominated flows and some types of transient problems. The results obtained by the developed scheme are then compared with the results generated by the other techniques in order to evaluate the applicability of the bubble function method. As discussed in this thesis the outcome from these comparisons proves the applicability of the bubble function method in solving realistic problems and its distinct superiority over traditional methods. Comparisons with field survey data representing pollutant dispersion in an estuary is particularly significant because this is the first time that such a comparison (i.e. experimental data) for the evaluation of this method has been carried out.

The conclusions of this research can be listed as:

- The bubble function method offers a practical solution technique which can replace upwinding based smoothing methods.

- Although the use of the bubble functions requires high order quadrature but this work shows that very high order numerical integration can be incorporated into ordinary finite element scheme without any difficulties.

- Its extension to multi dimensional problems is straightforward as tensor product approach can be used to construct two or three dimensional elements.

- It provides significant reduction in CPU time and computational effort in solving large scale industrially relevant problems.

- It can be extended to transient problems. This construction is also very significant because it offers a completely novel technique for temporal discretisation which has significant flexibility in comparison with often used θ or Taylor approaches.

5.2 Future Work

The most important extension of this work can be envisaged to be the construction of tensor product three dimensional bubble enriched elements. Although the basic aspects of such a method has been covered in this work the actual implementation of it has not been done. After construction of three dimensional elements they can be used to solve realistic problems in order to evaluate the performance of three dimensional bubble function schemes.

Bibliography

ALLAN, F.M. AND HAMDAN, M.H., 2002. Fluid mechanics of the interface region between two porous layers. *Applied Mathematics and computation*, **128**, pp 37-43.

ARIS, R., (1989). *Vectors, Tensors and the basic Equations of Fluid Mechanics.* Dover Publications, New York.

ARNOLD, D.N. BREZZI, F. AND FORTIN, M., 1984. A stable finite element for Stokes equation. *Calcolo*, **21**, pp 337-344.

BAI, W., 1997. The quadrilateral 'Mini' finite element for the stokes problem. *Computer methods in applied mechanics and engineering*, **143**, pp 41-47.

BAIOCCHI, C., BREZZI, F. AND FRANCA, L.P., 1993. Virtual bubbles and Galerkin-least-squares type methods. *Computer methods in applied mechanics and engineering*, **105**, pp 125-141.

BANK, R.E. AND WELFERT, B.D., 1990. A comparison between the mini-element and the Petrov-Galerkin formulations for the generalized stokes problem. *Computer methods in applied mechanics and engineering*, **83**, pp 61-68.

BATHE, K.J., 1996. *Finite Element Procedures*,. 2 edn. Prentice Hall, Englewood Cliffs, N.J.

BEAVERS, G.S. AND JOSEPH, D.D. 1967. Boundary conditions at the natural permeable wall. *Journal of Fluid mechanics*, **30/1**, pp 197-207

BIKANGAGA, J.H. AND NASSEHI, V., 1993. A mathematical model for the hydrodynamics and pollutants transport in long and narrow tidal rivers. *Applied Mathematical Modelling*, **17**, pp-415-422.

BREZZI, F., BRISTEAU, M., FRANCA, L.P., MALLET, M. AND ROGE, G., 1992. A relationship between stabilized finite element methods and the Galerkin method with bubble functions. *Computer methods in applied mechanics and engineering*, **96**, pp 117-129.

BREZZI, F., FRANCA, L.P., HUGHES, T.J.R. AND RUSSO, A. 1997. $b = \int g$. *Computer methods in applied mechanics and engineering*, **145**, pp 392-339.

BREZZI, F., FRANCA, L.P., AND RUSSO, A., 1998. Further considerations on residual-free bubbles for advective-diffusive equations. *Computer methods in applied mechanics and engineering*, **166**, pp 25-33.

BRINKMAN, H.C., 1947. A calculation of the viscous force exerted by a flowing fluid on a dense array of particles. *Applied Scientific Research*, **A1**, pp 27-34.

BURMAN, E. AND HANSBO, P., 2002. *A unified stabilized method for Stopkes' and Darcys' Equations.* Chalmer Finite Element Center University of Technology, Goteborg, Sweden.

DAS, D.B. AND NASSEHI, V., 2004. Validity of moving boundary finite element model for salt intrusion in a branching estuary. *Advances in Water Resources*, **27**, pp 725-735.

DONEA, J., 1984. A Taylor-Galerkin method for convection transport problems. *International Journal of Numerical Methods in Engineering*, **20**, pp 101- 120.

DONEA, J. AND HUERTA, A., 2003. *Finite element methods for flow problems.* John Wiley & Sons, Chichester.

DONEA, K., GIULIANI, S., LAVAL, H. AND QUARTAPELLE, L., 1984. Time-accurate solution of advection diffusion problems. *Computer methods in applied mechanics and engineering*, **45**, pp 123-146.

DONEA, J. ROIG, B. AND HUERTA, A., 2000. High order accurate time stepping schemes for convection-diffusion equation. *Computer methods in applied mechanics and engineering*, **182**, pp 392-339.

FORTIN, M., AND FORTIN, A., 1985. Newer Elements for incompressible Flow, Finite Element methods. *Wiley, New York* , **6**, pp171-187.

FRANCA, L.P., HUGHES, T.J.R. AND STENBERG, R., 1993. *Stabilized finite element methods for the Stokes problem. In: Nicolaides, R.A. and Gunzberger,* M.D..

Incompressible Fluid Dynamics-Trends and Advances. Cambridge University Press, Cambridge.

FRANCA, L.P. AND FARHAT, C., 1995. Bubble functions prompt unusual stabilized finite element methods. *Computer methods in applied mechanics and engineering*, **123**, pp 299-308.

FRANCA, L.P. AND RUSSO, A., 1996. Deriving upwinding, mass lumping and selective reduced integration by residual free bubbles, *Applied Mathematics Letters*, **9**, pp 83-88.

FRANCA, L.P. AND RUSSO, A., 1997a. Mass lumping emanating from residual free bubbles. *Computer methods in applied mechanics and engineering*, **142**, pp 353-360.

FRANCA, L.P. AND RUSSO, A., 1997b. Unlocking with residual-free bubbles. *Computer methods in applied mechanics and engineering*, **142**, pp 361-364.

FRANCA, L.P., FARHAT, C., MACEEDO, A.P.AND LESOINNE, M., 1997. Residual free bubbles for Helemholtz equation. *Computer methods in applied mechanics and engineering*, **40**, pp 4003-4009.

GRAVIMIER, V., WALL, W.A., AND RAMM, E., 2004. A three-level finite element method for the in stationary incompressible Navier-Stokes equations. *Computer methods in applied mechanics and engineering*, **193**, pp 1323-1366.

GERALD, C.F. AND WHEATLEY, P.O., 1984. *Applied numerical analysis*. Addison-Wesley.

GUNN, D.J. AND YENIGUN, O., 1985. Modelling of tidal motion in shoaling waters; the Estuary of Milford Haven. *Estuarine, Coastal and Shelf Science*, **21**, pp 337-356.

HUGHES, T.J.R., 1995. Multi scale phenomena, Green's functions, the Dirichlet – to-Neumann formulation, sub-grid scale models, bubbles and the origins of stabilized methods. *Computer methods in applied mechanics and engineering*, **127**, pp 381-401.

HUGHES, T.J.R. AND STEWART, J., 1996. A space time formulation for multiscale phenomena. *Computer methods in applied mechanics and engineering*, **74**, pp 217-229.

HUGHES, T.J.R, FEIJÓO, G.R., MAZZEI, L. AND QUINCY, J,-B., 1998, The Variational multi-scale method a paradigm for computational mechanics. *Computer methods in applied mechanics and engineering*, **166**, pp 3-24.

HSU, C.T. AND CHENG, P., 1985. The Brinkman model for natural convection about a semi-infinite vertical flat plate in a porous medium. *International journal of heat and mass transfer*, **28**, pp 683-697.

IRONS, B.M., 1970. A frontal solution for finite element analysis. *International Journal for Numerical Methods in Engineering*, **2**, pp 5-32.

JENSON, V.G. AND JEFFREYS, G.V., 1983. *Mathematical Methods in Chemical Engineering*. Academic Press, London.

LAMBERT, J.D. 1993. *Numerical Methods of Ordinary Differential Systems*. Wiley, New York.

LAYTON, W.J., SCHIEVECK, F. AND YOTOV, I., 1993. Coupling fluid flow with porous media flow. *SIAM Journal of numerical analysis*, **40/6**, pp 2195-2218.

KAFAI, A. AND NASSEHI, V., 1999. Modelling of salt intrusion in the Upper Milford Haven Estuary- A branching tidal water system. *Proceedings WATERMATIX -2000*, **1**, pp 67-74.

KAVIANY, M., 1986. Non-Darcian effects on natural convection in porous media confined between horizontal cylinders. *International journal of heat and mass transfer*, **29**, pp 1513-1519.

KING, S.A. AND NASSEHI, V., 1991. Finite element methods for the convection diffusion equation. *Islamic Republic of Iran Journal of Engineering*, **4/3-4**, pp 93-100.

KIM, Y., AND LEE,.S., 1998. Modified Mini finite element for the Stokes problem in $\Re^2$ and $\Re^3$. *Advance in Computation Mathematics*, **12**, pp 261-272.

NIELD, D.A. AND BEJAN, A., (1992). *Convection in porous media*. Springer-Verlag, New York.

NELSON-SMITH, A., 1965. Marine biology of Milford Haven: physical environment. *Field studies*, **2**, pp 155-188.

NASSEHI, V., 1998. Modelling of Combined Navier-Stokes and Darcy Flows in Crossflow Membrane Filtration., *Chemical Engineering Science*, **53/6**, pp 1253-1265.

NASSEHI, V., 2002. *Practical aspects of finite element modelling of polymer processing.* John Wiely & Sons, Chichester.

NASSEHI, V., 1981. *Numerical modelling of tidal dynamics in a one dimensional branching estuary.* PhD thesis, University of Wales Swansea.

NASSEHI, V. AND DAS, D.B., 2007. *Computational methods in the Management of Hydro-Environmental Systems.* IWA Publishing, London.

NASSEHI, V. AND WILLIAMS, D.J.A, 1986. Mathematical model of upper Milford Haven a branching estuary. *Estuarine, Coastal and Shelf science*, **23**, pp 403-418.

NIELD, D.A. AND BEJAN, A., 1992. *Convection in porous media.* Springer-Verlag, New York Inc.

OKUMURA, H., AND KAWAHARA, M., 2003, A new stable Bubble Element for incompressible Fluid Flow Based on a Mixed Petrov-Galerkin Finite Element Formulation. *International journal of computational fluid dynamics*, **17/4**, pp 275-282

PARVAZINIA, M., NASSEHI, V., WAKEMAN, R.J. AND GHOREISHY, M.H.R., 2006. Finite element modelling of flow through a porous medium between two parallel plates using the Brinkman equation, Transport in porous media, In press.

PARVAZINIA, M., NASSEHI, V. AND WAKEMAN, R.J., 2006a. Multi-scale finite element modelling of laminar steady flow through highly permeable porous media. *Chemical Engineering Science*, **61**, pp 586-596.

PARVAZINIA, M., NASSEHI, V. AND WAKEMAN, R.J., 2006b. Multi-scale finite element modelling using bubble function method for a convection-diffusion problem. *Chemical Engineering Science*, **61**, pp 2742-2751.

PIERRE, R., 1988. Simple C^0 approximation for the computation of incompressible Flows. *Computer methods in applied mechanics and engineering*, **68**, pp 205-227

PITTMAN, J.F.T., 1989. *Finite elements for field problems. In: Tucker III, C.L., Computer modelling for polymer processing*, Hanser Publishers, Munich.

REDDY, J.N., 1993. *An introduction to the Finite Element Method,* 2 EDN.,McGraw-Hill, New York.

REDDY, J.N., 1986. *Applied functional analysis and variational methods in engineering.* McGraw-Hill, New York.

ROACHE, P.J., 1972. *Computational Fluid Dynamics.* Hermosa Publishers, Alberquerque, N.M.

WEST, J.R., 1978. *A report on the hydrodynamics and water quality of Upper Milford Haven.* Department of Civil Engineering, University of Birmingham, Vol. 1 and Vol. 2.

WEST, J.R., 1979. *A report on the hydrodynamics and water quality of Upper Milford Haven.* Department of Civil Engineering, University of Birmingham, Vol. 3.

WEST, J.R. AND BROYD, T.W., (1981). Dispersion coefficient in Estuaries. *Processing Institution Civil Engineers,* **72/2**, pp721-737.

ZIENKIEWICZ, O.C. AND TAYLOR, R.L., 1994. *The finite element method.* McGraw-Hill, London.

Appendix A
Publications

I. Paper 1

Multi-scale finite element modelling of flow through porous media with curved and contracting boundaries to evaluate different types of bubble functions

V. Nassehi[1], M. Parvazinia[2], A.Khan[1]

1-Department of Chemical Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK

2- Department of Mathematical Modelling and Simulation, Iran Polymer and Petrochemical Institute, Tehran, Iran, P.O.Box: 14185/458

ABSTRACT

The Brinkman equation is used to model the isothermal flow of the Newtonian fluids through highly permeable porous media. Due to the multi-scale behaviour of this flow regime the standard Galerkin finite element schemes for the Brinkman equation require excessive mesh refinement at least in the vicinity of domain walls to yield stable and accurate results. To avoid this, a multi-scale finite element method is developed using bubble functions. It is shown that by using bubble enriched shape functions the standard Galerkin method can generate stable solutions without excessive near wall mesh refinements. In this paper the performances of different types of bubble functions are evaluated. These functions are used in conjunction with bilinear Lagrangian elements to solve the Brinkman equation via a penalty finite element scheme.

Keywords: Finite element, Multi-scale method, Porous media, Newtonian fluid flow, Bubble function, Static condensation.

II. Correspondence related to the Paper 1

Report on

"Multi-scale finite element modelling of flow through porous media with curved and contracting boundaries to evaluate different types of bubble functions"

by V. Nassehi et al

In the paper, numerical methods are considered for the porous media flow under the Brinkman's law with constant permeability. Different practical bubble functions are used to enrich the finite element space mainly to achieve the stability on a coarse mesh. A penalty method is used to deal with the incompressible condition. Numerical experiments are also presented. The topic has certain importance and the style is suitable for CiCP.

I recommend the paper with some modifications.

Some minor suggestions are listed below.

1) The penalty method is used in (54)-(55). But the authors didn't discuss about how to choose the penalty parameter λ_0, nor reported it in all the numerical experiments. The authors reported the results on the pressure, but nothing related to the pressure recovery process was presented in the paper.

2) In the second numerical experiment, the domain has a curved boundary, but not explicitly given. Can the method be used for any kind of curved domain? If so, please explain it. If not, indicate the special form of the boundary.

From: "Commun. Comput. Phys." <cicp@global-sci.com>

To: "Vahid Nassehi" <V.Nassehi@lboro.ac.uk>

Cc: "Commun. Comput. Phys." <cicp@global-sci.com>

Sent: Thursday, November 09, 2006 4:13 AM

Subject: Re: Review CiCP 06-71

Dear Prof. Nassehi,

Thanks again for revising your paper and the referee found your revision satisfactory. I am pleased to inform you that your paper is now accepted for publication in Communications in Computational Physics (CiCP).

1. Please send in your files (in .zip or .tar formats) to the editorial office of CiCP at cicp@global-sci.com.

2. Please also fill in the copyright form in
http://www.global-sci.com/authors/copyright.pdf
and email me back the completed form. You may also fax it to my office at (852) 3411 5811.

Thank you again for submitting your paper to CiCP and we look forward to receiving your other submissions in the near future.

Sincerely,

Tao

Tao Tang

Managing Editor

Communications in Computational Physics

Communications in Computational Physics is available electronically at
http://www.global-sci.com

III. Paper 2

Submitted to Communications in Computational Physics :

A novel multi-scale finite element time discretisation method for transient transport phenomena using bubble functions

V. Nassehi[1], M. Parvazinia[2], A.Khan[1]

1-Department of Chemical Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK

2- Department of Mathematical Modelling and Simulation, Iran Polymer and Petrochemical Institute, Tehran, Iran, P.O.Box: 14185/458

ABSTRACT

A novel time discretisation method for the transient transport phenomena is proposed. The essential feature of this method is that is allows the treatment of initial value problem as a boundary value problem. Therefore it is possible to use a similar strategy for both spatial and temporal discretisation of the field variables. The technique is used to model transient diffusion and transient convection-diffusion equations. Cases where the transient response behaves in a multi-scale manner are studied and therefore, bubble functions are used to develop a multi-scale finite element scheme. The phenomenon of the proposed scheme is compared with the widely used theta time stepping method. The numerical results are obtained by the proposed scheme are validated by comparison with the analytical solution.

Keywords: Transient transport phenomena; Bubble function; Multi-scale; Finite element; Theta time stepping method.

Appendix B
Program

```
C...........................................................
C                                                          .
C   GENERAL SPECIFICATIONS :                               .
C                                                          .
C      - CONTINUOUS OR  DISCRETE PENALTY METHODS FOR FLOW EQ.  .
C        IN CYLINDRICAL AND CARTESIAN COORDINATE SYSTEMS   .
C      - BUBNOV-GALERKIN OR  STREAMLINE UPWIND/PETROV-GALERKIN .
C        FOR HEAT EQ.                                      .
C      - IMPLICIT THETA TIME STEPPING OR TAYLOR-GALERKIN   .
C        FOR TRANSIENT PROBLEMS                            .
C      - POWER-LAW , CARREAU AND CEF MODELS                .
C      - PRESSURE BOUNDARY CONDITION                       .
C      - MODELLING OF SLIP PHENOMENON USING LAGRANGE MULTIPLIER .
C        OR DIRECT INCORPORATION OF NAVIER'S SLIP CONDITION .
C      - FRONTAL SOLVER ( NON-SYMMETRIC )                  .
C      - SIMULATION OF FREE SURFACE FLOW USING A PSEUDO-DENSITY .
C        METHOD                                            .
C      - SOLUTION OF CONCENTRATION ( CONVECTIVE TRANSPORT ) .
C        EQUATION                                          .
C      - CALCULATION OF THE EFFECTIVE FILLER VOLUME FRACTION .
C        ( IN RUBBER/CB COMPOUNDS )                        .
C      - MODELING OF THE FIXED OR MOVING MESH FLOW PROBLEMS .
C                                                          .
C                                                          .
C        THE ORIGINAL PROGRAM IS WRITTEN BY                .
C                    M.H.R.GHOREISHY                       .
C                                                          .
C        LATER REVISED AND EXTENDED TO PRESENT FORM        .
C        WHICH USES BUBBLE FUNCTION METHOD                 .
C        OCT 2003 TO JUN 2006                              .
C                                                          .
C                                                          .
C   THIS PROGRAM IS PRIMARILY DESIGNED TO SIMULATE RUBBER  .
C   MIXING IN PARTIALLY FILLED BATCH INTERNAL MIXERS.      .
C                                                          .
C              EXTENDED FORTRAN 77                         .
C...........................................................
C   LIST OF SYMBOLS                                        .
C...........................................................
C
C   AINIT . . . INITIAL CONDITION IN OLD CONFIGURATION
C               ( MODELLING OF MOVING MESH FLOW PROBLEMS [MMFP] )
C   AK. . . . . THERMAL CONDUCTIVITY
C   BC. . . . . PRIMARY B.C. ARRAY
C   BCT . . . . PRIMARY B.C. ARRAY   ( TEMPERATURE )
C   BCV . . . . PRIMARY B.C. ARRAY   ( VELOCITY    )
C   BCS . . . . PRIMARY B.C. ARRAY   ( STREAM FUNCTION ) STEADY STATE
PROBLEMS
C   BCF . . . . PRIMARY B.C. ARRAY   ( FREE SURFACE    )
C   BCC . . . . PRIMARY B.C. ARRAY   ( CONCENTRATION   )
```

```
C   CINIT . . . INITIAL CONCENTRATION
C   CORP. . . . NODAL COORDINATE VAUES ( OLD CONFIGURATION )
C   CP. . . . . HEAT CAPACITY
C   CRF . . . . CONCENTRATION
C   CRO . . . . CONCENTRATION ( INITIAL OR PREVIOUS TIME STEP VALUES )
C   CPHI. . . . EFFECTIVE FILLER VOLUME FRACTION (EFVF)
C   CPHIO . . . INITIAL VALUE OF EFVF
C   CPHIA . . . INITIAL ESTIMATE FOR EFVF
C   DMASS . . . MASS MATRIX
C   DT. . . . . TIME STEP
C   ELF . . . . ELEMENTAL LOAD VECTOR
C   ELF1. . . . ELEMENTAL LOAD VECTOR VECTOR ( PREVIOUS  TIME STEP )
C   ELSTIF. . . ELEMENTAL STIFFNESS MATRIX
C   ELSTID. . . ELEMENTAL STIFFNESS MATRIX   ( PREVIOUS TIME STEP  )
C   ELVIS . . . ELONGATIONAL VISCOSITY
C   EQ. . . . . A VARIABLE USED IN FRONTAL ROUTINE ( GLOBAL MATRIX )
C   GAUSS . . . GAUSS POINTS ARRAY
C   GF. . . . . VELOCITY - CURRENT SOLUTION VECTOR
C   GFA . . . . VELOCITY - INITIAL VECTOR
C   GFI . . . . VELOCITY - INITIAL VECTOR (2)
C   GFM . . . . MESH VELOCITY  (MMFP)
C   GFS . . . . FREE SURFACE LOCATION FUNCTION
C   GFSI. . . . FREE SURFACE LOCATION FUNCTION ( INITIAL )
C   GFSO. . . . FREE SURFACE LOCATION FUNCTION ( PREVIOUS TIME STEP )
C   SRF . . . . STREAM FUNCTION ( STEADY STATE PROBLEMS )
C   IF. . . . . FULL INTEGRATION INDEX
C   IR. . . . . REDUCED INTEGRATION INDEX
C   IVIS. . . . VISCOSITY EQUATION:
C                         1 : POWER-LAW ;
C                         2 : CARREAU
C   JMOD. . . . A VARIABLE USED IN FRONTAL ROUTINE
C   LDEST . . . A VARIABLE USED IN FRONTAL ROUTINE
C   LHED. . . . A VARIABLE USED IN FRONTAL ROUTINE
C   LPIV. . . . A VARIABLE USED IN FRONTAL ROUTINE
C   MAXFR . . . SIZE OF ARRAYS IN FRONTAL ROUTINE
C   MDF . . . . NO. OF D.O.F. AT EACH NODE
C   MDFT. . . . NO. OF D.O.F. AT EACH NODE ( TEMPERAT ) , ARRAY
C   MDFV. . . . NO. OF D.O.F. AT EACH NODE ( VELOCITY ) , ARRAY
C   MDFS. . . . NO. OF D.O.F. AT EACH NODE ( STREAM FUNCTION ) , ARRAY
C   MDFF. . . . NO. OF D.O.F. AT EACH NODE ( FREE SURFACE ) , ARRAY
C   MDFC. . . . NO. OF D.O.F. AT EACH NODE ( CONCENTRATION/EFVF#) , ARRAY
C   MRCL. . . . RECORDED FILE LENGTH (DEFAULT) FOR FRONTAL ROUTINE
C   NBD1. . . . NO. OF FIRST TYPE B.C. NODES
C   NBF . . . . NO. OF THIRD TYPE B.C. ELEMENTS
C   NCARB . . . =1 DO NOT NO CALCULATE CONCENTRATION =2 CALCULATE
CONCENTRATION
C   NCOD. . . . CODE FOR PRIMARY B.C.
C   NCODT . . . CODE FOR PRIMARY B.C. ( TEMPERAT   ) , ARRAY
C   NCODV . . . CODE FOR PRIMARY B.C. ( VELOCITY   ) , ARRAY
C   NCODS . . . CODE FOR PRIMARY B.C. ( STREAM FUNCTION ) ,ARRAY
C   NCODF . . . CODE FOR PRIMARY B.C. ( FREE SURFACE    ) ,ARRAY
C   NCODC . . . CODE FOR PRIMARY B.C. ( CONCENTRATION   ) ,ARRAY
C   NCYL   . . . RHEOLOGICAL EQUATION AND COORDINATE SYSTEM TYPE:
C                    1 : GENERALIZED NEWTONIAN (X,Y)
C                    2 : CEF   (R,THETA)
```

```
C                    3 : GENERALIZED NEWTONIAN (R,Z), AXISYMMETRIC
C    NDFT. . . . NO. OF D.O.F. AT EACH NODE ( TEMPERAT ) ( COMPARE WITH
MDFT)
C    NDFV. . . . NO. OF D.O.F. AT EACH NODE ( VELOCITY ) ( COMPARE WITH
MDFV)
C    NDFS. . . . NO. OF D.O.F. AT EACH NODE ( STRM FC. ) ( COMPARE WITH
MDFS)
C    NDFF. . . . NO. OF D.O.F. AT EACH NODE ( FREE SURFACE) ( COMPARE WITH
MDFF)
C    NDFC. . . . NO. OF D.O.F. AT EACH NODE ( CONCENTRATION ( COMPARE WITH
MDFF)
C    NDNT. . . . TOTAL NO. OF D.O.F. AT EACH ELEMENT ( TEMPERAT  )
C    NDNV. . . . TOTAL NO. OF D.O.F. AT EACH ELEMENT ( VELOCITY  )
C    NDNS. . . . TOTAL NO. OF D.O.F. AT EACH ELEMENT ( STREAM FUNCTION )
C    NDNF. . . . TOTAL NO. OF D.O.F. AT EACH ELEMENT ( FREE SURFACE )
C    NDNC. . . . TOTAL NO. OF D.O.F. AT EACH ELEMENT ( CONCENTRATION)
C    NELM. . . . MAX. NO. OF ELEMENTS
C    NEM . . . . NO. OF ELEMENTS
C    NEQ . . . . NO. OF VELOCITY FIELD EQUATIONS
C    NET . . . . NO. OF TEMPERATURE FIELD EQUATIONS
C    NES . . . . NO. OF STREAM FUNCTION FIELD EQUATIONS
C    NEF . . . . NO. OF FREE SURFACE FUNCTION EQUATIONS
C    NEC . . . . NO. OF CONCENTRATION/EFVF EQUATIONS
C    NFREE . . . = 1 FULLY FILLED
C                = 2 FREE SURFACE MODELLING ( WITHOUT AIR COMPRESSIBILITY )
C                = 3 FREE SURFACE MODELLING ( WITH AIR COMPRESSIBILITY AT
HIGH
C                                                  PRESSURE REGIONS )
C    NITER . . . MAX. NO. OF PERMITTED ITERATIONS
C    NITNS . . . INITIAL DATA OPTIONS
C                = 1 INITIAL VARIABLES INPUTED VIA AN NIPUT FILE
C                = 2 INITIAL VARIABLES ARE FOUND VIA A PREVIOUSLY
C                  EXECUTED MODEL (FIXED MESH)
C                = 3 INITIAL VARIABLES ARE FOUND VIA A PREVIOUSLY
C                  EXECUTED MODEL (MOVING MESH)
C    NK. . . . . A VARIABLE FOR FRONTAL ROUTINE
C    NNEE. . . . NUMBERS OF ELEMENTS ATTACHED TO EACH NODE , ARRAY
C    NNISO . . . =1 ISOTHERMAL ,  =2  NONISOTHERMAL
C    NNM . . . . NO. OF NODES
C    NNOD. . . . MAXIMUM NO. OF NODES
C    NOD . . . . ELEMENT CONNECTIVITY MATRIX
C    NOP . . . . ELEMENT CONNECTIVITY MATRIX ( FOR PRE-FRONT)
C    NODPV . . . ELEMENT CONNECTIVITY ( OLD CONFIGURATION )
C    NOPP. . . . CODED VALUE OF THE FIRST D.O.F. AT EACH NODE
C    NOPPT . . . CODED VALUE OF THE FIRST D.O.F. AT EACH NODE ( TEMPERAT )
C    NOPPV . . . CODED VALUE OF THE FIRST D.O.F. AT EACH NODE ( VELOCITY )
C    NOPPS . . . CODED VALUE OF THE FIRST D.O.F. AT EACH NODE ( STREAM
FUNC.)
C    NOPPF . . . CODED VALUE OF THE FIRST D.O.F. AT EACH NODE ( FREE SURFACE
)
C    NOPPC . . . CODED VALUE OF THE FIRST D.O.F. AT EACH NODE ( CONC./EFVF
)
C    NPE . . . . NUMBER OF NODES PER ELEMENTS
C    NSDOF . . . A 4 COLUMN ARRAY. COLUMNS 1-3 INDICATE THE SLIP SIDE AND
C                COLUMN 4 INDICATES THE NO. OF SLIP ELEMENT
```

```
C    NSIZ. . . . SIZE OF THE GLOBAL VECTOR OF UNKNOWNS
C    NSTF. . . . SIZE OF ELEMENT STIFFNESS MATIX AND LOAD VECTOR
C    NSUPG . . . TYPE OF UPWINDING USED IN HEAT EQUATION
C                =1 STANDARD GALERKIN METHOD ( NO UPWINDING )
C                =2 STREAMLINE UPWINDING
C                =3 STREAMLINE UPWINDING/PETROV-GALERKIN
C    NTIME . . . NO. OF TIME STEPS
C    NTRAN . . . =0  STEADY STATE  ,   NE.0  TRANSIENT
C    NWR . . . . NO. OF SAMPLE NODES FOR RECORDING TRANSIENT SOLUTIONS IN
C                FILE 'TRANS.OUT'
C    PBU . . . . BOUNDARY PRESSURE (2ND DOF)
C    PMG . . . . GLOBAL LUMPED MASS MATRIX USED IN VARIATIONAL RECOVERY
C                FORMULATION
C    PPVL. . . . CHARACTERISTIC LENGTH OF SLIP EQUATION
C    PRHS. . . . RIGHT HAND SIDE OF THE GLOBAL VECTOR USED IN VARIATIONAL
C                RECOVERY METHOD
C    PSE9. . . . PRESSURE AT EACH NODE FOUND USING VARIATIONAL RECOVERY
METHOD
C    PSEE. . . . PRESSURE AT EACH NODE FOUND USING LOCAL AVERAGING TECHNIQUE
C    PSIC. . . . A PARAMETER IN CEF EQUATION
C    PSLIM . . . PRESSURE LIMIT WHICH ABOVE THAT AIR COMPRESS. SHOULD BE
APPLIED
C    PSOUT . . . ARRAY OF PRESSURE VALUES AT EACH NODE
C    PVKOL . . . A VARIABLE FOR FRONTAL ROUTINE
C    QQ. . . . . A VARIABLE FOR FRONTAL ROUTINE
C    RELVIS. . . RELATIVE VISCOSITY ( DEFINED AS THE RATIO OF THE COMPOUND
C                VISCOSITY TO THE GUM RUBBER VISCOSITY )
C    R1. . . . . A VARIABLE FOR FRONTAL ROUTINE
C    SO. . . . . VELOCITY - PREVIOUS TIME STEP VECTOR
C    STIME . . . STARTING TIME
C    TF. . . . . TEMPERATURE - CURRENT SOLUTION VECTOR
C    TFA . . . . TEMPERATURE - INITIAL VECTOR
C    TFI . . . . TEMPERATURE - INITIAL VECTOR (2)
C    THETA . . . VALUE OF THETA IN IMPLICIT-THETA TIME STEPPING METHOD
C    TIME. . . . TIME
C    TINIT . . . INITIAL TEMPERATURE
C    TN. . . . . ERROR NORM FOR TEMPERATURE
C    TO. . . . . TEMPERATURE - PREVIOUS TIME STEP VECTOR
C    TOLE1 . . . TOLERANCE VALUE FOR NONLINEAR SOLUTION SCHEME  1
C    TOLE2 . . . TOLERANCE VALUE FOR NONLINEAR SOLUTION SCHEME  2
C    TOLE3 . . . TOLERANCE VALUE FOR NONLINEAR SOLUTION SCHEME  3
C    TTXX. . . . NORMAL STRESS (XX)
C    TTXY. . . . SHEAR STRESS (XY)
C    TTYY. . . . NORMAL STRESS (YY)
C    VHS . . . . VELOCITY GRADIENT COMPONENTS
C    VINIT . . . INITIAL VELOCITY
C    VN. . . . . ERROR NORM FOR VELOCITY
C    VPROP . . . PHYSICAL AND RHEOLOGICAL DATA (ARRAY)
C                1  . . . POWER LAW CONSTANT ( CONSISTENCY )
C                2  . . . POWER LAW INDEX
C                3  . . . TEMPERATURE SENSITIVITY
C                4  . . . REFERENCE TEMPERATURE
C                5  . . . RELAXATION TIME
C                6  . . . SLIP COEFFICIENT
```

```
C               7  . . . CHARACTERISTIC CONS.(FILLER DISPERSION KINETIC EQ.
)
C               8  . . . PROPERTIES OF VOID REGIONS ( VISCOSITY )
C               9  . . . PROPERTIES OF VOID REGIONS ( DENSITY )
C               10 . . . PROPERTIES OF VOID REGIONS ( CONDUCTIVITY )
C               11 . . . PROPERTIES OF VOID REGIONS ( HEAT CAPACITY )
C               12 . . . CHARACTERISTIC CONSTANT FOR THE DEFINITION OF PSIC
C               13 . . . CHARACTERISTIC CONSTANT FOR THE DEFINITION OF PSIC
C               14 . . . MATERIAL DENSITY
C               15 . . . PENALTY PARAMETER
C               21 . . . CONSTANT USED TO RELATE EFVF AND VISCOSITY
C               22 . . . CONSTANT USED TO RELATE EFVF AND VISCOSITY
C   VRES. . . . VELOCITY ( MAGNITUDE OF THE VELOCITY VECTOR )
C   WT. . . . . WEIGHTS ARRAYS FOR GAUSSIAN QUADRATURE
C   X . . . . . X-COORDINATE ( CARTESIAN )
C   Y . . . . . Y-COORDINATE ( CARTESIAN )
C
C
C   NOTE : ANY SYMBOL NOT DEFINED IN THE ABOVE LIST IS LOCALLY
C          DEFINED
C.............................................................
C.............................................................
C  LIST OF SUBROUTINES                                        .
C.............................................................
C.............................................................
C ADDELF. . ADD ELEMENT LOAD VECTOR FOR TRANSIENT ANALYSIS
C ADDSF  . . ADD STIFFNESS MATRICES FOR TRANSIENT ANALYSIS
C ANODAE. . FIND THE CONNECTIVITY OF EACH NODE TO ADJACENT ELEMENTS
C ARR2ZF. . INITIALIZE TWO DIMENSIONAL ARRAYS
C ARRZRF. . INITIALIZE ONE DIMENSIONAL ARRAYS
C ARRZRI. . INITIALIZE ONE DIMENSIONAL ARRAYS ( INTEGER )
C BACSUB. . BACK SUBSTITUTION FOR FRONTAL ALGORITHM
C BOUN01. . RESTRICTS A VARIABLE BETWEEN 0 AND 1
C BUINTG. . BOUNDARY INTEGRAL FOR FLOW EQUATIONS
C CAR2CYL . TRANSFORMS THE COORDINATES FROM CARTESIAN TO CYLINDRICAL
C CARBON. . SOLUTION OF CONCENTRATION EQUATION
C COBET . . CALCULATION OF BETA  IN UPWINDING FORMULATION
C COEFF . . CALCULATION OF ALPHA IN UPWINDING FORMULATION
C COORD . . FIND THE GLOBAL COORDINATE
C CYL2CAR . TRANSFORMS THE COORDINATES FROM CYLINDRICAL TO CARTESIAN
C DAKE. . . DETERMINATION OF LOCAL COORDINATES OF A NODE
C EFVF. . . SOLUTION OF EFFECTIVE FILLER VOLUME FRACTION EQUATION
C ELFC. . . CALCULATION OF THE ELEMENTAL LOAD VECTOR FOR TRANSIENT ANALYSIS
C ELFT. . . CALCULATION OF THE TRANSIENT ELEMENTAL LOAD VECTOR FOR HEAT
EQUATION
C ELFTGL. . CALCULATION OF THE STIFFNESS AND LOAD VECTOR MATRICES IN
C           TAYLOR-GALERKIN METHOD
C ELFTS . . CALCULATION OF THE ELEMENTAL LOAD VECTOR FOR HEAT EQUATION
C FILTER. . APPROXIMATION OF F VALUE ( FREE SURFACE LOCATION FUNCTION )
C FLCYL . . SOLUTION OF THE FLOW EQUATION ( CONT. PENALTY AND CYL.
COORDINATE )
C FLOWCN. . SOLUTION OF THE FLOW EQUATION ( CONTINIOUS PENALTY METHOD )
C FPSL  . . INTERPOLATION OF F VALUE IN AN ELEMENT
C FRONT . . FRONTAL SOLVER
C FRSFVL. . CALCULATION OF F VALUE FOR FREE SURFACE ANALYSIS
```

```
C GJE . . . GAUSS-JORDAN ELIMINATION
C HGSTVL. . FIND THE MAXIMUM AND MINIMUM VALUES OF CALCULATED PRESSURE
C           AND TEMPERATURE
C JACCYL. . CALCULATION OF THE JACOBIAN MATRIX AND ITS DETERMINANT (CYL.)
C JACOB2. . CALCULATION OF THE JACOBIAN MATRIX AND ITS DETERMINANT
C LAGSH1. . 1D SHAPE FUNCTIONS
C LUMP. . . LUMPINFG OF THE MASS MATRIX
C MASCYL. . MASS MATRIX CALCULATION ( CYLINDRICAL COORDINATE )
C MASS. . . MASS MATRIX CALCULATION
C MASST . . MASS MATRIX CALCULATION
C MASSW . . MASS MATRIX CALCULATION
C MSHUPD. . UPDATING MESH IN PURE LAGRANGIAN METHOD
C MSHINI. . FIND THE INITIAL VALUE FOR INTERPOLATION PURPOSES
C MULT2 . . MATRIX MULTIPLICATION
C MULT3 . . MATRIX MULTIPLICATION
C OUTPUT. . WRITE OUTPUT
C PREFNT. . PRE-FRONT ROUTIE
C PRSCYL. . CALCULATION OF THE PRESSURE (VARIATIONAL RECOVERY IN
CYLINDRICAL
C           COORDINATE SYSTEM )
C PRESD . . CALCULATION OF PRESSURE IN DISCRETE PENALTY METHOD
C PRESS . . CALCULATION OF THE PRESSURE BASED ON THE VARIATIONAL RECOV.
METHOD
C PUBCLV. . IMPOSITION OF THE BOUNDARY CONDITION FOR FLOW EQ. (CYLINDRICAL)
C PUTBCV. . IMPOSITION OF THE BOUNDARY CONDITION FOR FLOW EQUATION
C PVRGMX. . GLOBAL LUMPED MASS MATRIX FOR PRESSURE CALCULATION
C QHT . . . CALCULATION OF THE GENERATED HEAT BASED ON THE VISCOUS EFFECT
C RENWAL. . RENEWING OF THE SOLUTION VECTOR USING OVER-RELAXATION METHOD
C RSAVE . . LEFT HAND SIDE EQUAL TO THE RIGHT HAND SIDE
C RSAVI . . LEFT HAND SIDE EQUAL TO THE RIGHT HAND SIDE  (INTEGER)
C RTDER . . CALC. OF SHAPE FUNCTION DERIVATIVES IN CYLINDRICAL COORDINATE
C RTFIND. . INTERPOLATES AND FINDS THE RADUIS AND ANGLE INSIDE AN ELEMENT
C SH9DD . . CALCULATION OF THE HIGHER ORDER DERIVATIVES OF THE SHAPE
FUNCTIONS
C           IN LOCAL COORDINATES
C SHAPE . . SHAPE FUNCTIONS AND THEIR DERIVATIVES
C STICYL. . STIFFNESS MATRIX CALCULATION  ( CYLINDRICAL COORDINATE )
C STIFD . . STIFFNESS MATRIX CALCULATION  ( DISCRETE    )
C STIFF . . STIFFNESS MATRIX CALCULATION  ( CONTINUOUS )
C STIFFS. . STIFFNESS MATRIX CALCULATION  ( FLOW )
C STIFFT. . STIFFNESS MATRIX CALCULATION  ( HEAT )
C STRMFC. . STREAMFUNCTION CALCULATION
C TMPRUR. . TEMPERATURE CALCULATION
C TMPTGL. . SOLUTION OF THE HEAT EQUATION USING TAYLOR-GALERKIN METHOD
C TRAP. . . TRANSPOSE A MATRIX
C UNITVC. . CALCULATION OF THE UNIT VECTORS
C UPWIND. . CALCULATION OF THE UPWIND MULTIPLIER
C UTNML . . CALCULATION OF THE COMPONENTS OF UNIT VECTOR NORMAL TO THE
BOUNDARY
C UVN . . . CALCULATION OF THE VELOCITY AT ANY ELEMENT INTERIOR POINT
C UVN1D . . CALCULATION OF THE VELOCITY AT ANY ELEMENT BOUNDARY POINT
C UWPARA. . CALCULATION OF THE "DONEA" PARAMETERS FOR UPWINDING
C VCA2CL. . TRANSFORMATION OF VECTOR COMPONENTS TO CYLINDRICAL
C VCL2CA. . TRANSFORMATION OF VECTOR COMPONENTS TO CARTESIAN
C VISCEF. . CALCULATION OF THE VISCOSITY ( CEF IN CYLINDRICAL COORDINATE )
```

```
C VISCOS. . CALCULATION OF THE SHEAR RATE AND VISCOITY
C VISEQU. . VISCOSITY CALCULATION
C VISRHD. . CALCULATION OF THE VELOCITY GRADIENTS USING VARIATIONAL
RECOVERY
C VNORM . . CALCULATION OF THE ERROR NORM
C.....................................................................
..
C     INCLUDE 'SIZEF'
      PARAMETER ( NELM  = 40000    )
      PARAMETER ( NNOD  = 160000   )
      PARAMETER ( NSTF  = 18       )
      PARAMETER ( NSIZ  = 400000   )
      PARAMETER ( MAXFR = 2500     )
      PARAMETER ( MRCL  = 10024    )
C
C....  NELM   MAXIMUM NO. OF ELEMENTS
C....  NNOD   MAXIMUM NO. OF NODES
C....  NSTF   SIZE OF ELEMENT STIFFNESS MATIX AND LOAD VECTOR
C....  NSIZ   SIZE OF GLOBAL UNKNOWNS
C....  MAXFR  SIZE OF ARRAYS IN FRONTAL SUB.
C....  MRCL   SIZE OF RECORD LENGTH FOR FRONTAL ROUTINE FILE ( TO FIX THE
C             SCRATCH FILE CORRECTLY
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C.... STRING VARIABLE DECLARATION
C
      CHARACTER TITLE*60
      CHARACTER CU*2       , CV*2       , CT*2
      CHARACTER CF*2       , CC*2
      CHARACTER FNAME1*30 , FNAME2*30 , FNAME3*30
C
C.... STORAGE ALLOCATION
C
      DIMENSION NOD    ( NELM , 9 )
      DIMENSION NOP    ( NELM , 9 )
      DIMENSION X      ( NNOD ) , Y      ( NNOD )
C.............
      DIMENSION CORP   ( NNOD , 2 )
      DIMENSION NODPV  ( NELM , 9 )
      DIMENSION AINIT  ( NSIZ , 5 )
C.............
      DIMENSION GF     ( NSIZ ) , GFI    ( NSIZ ) , GFA ( NSIZ ) ,
     1          TF     ( NSIZ ) , TFI    ( NSIZ ) , TFA ( NSIZ ) ,
     1          GFS    ( NSIZ ) , GFSI   ( NSIZ ) , GFSO( NSIZ ) ,
     1          THF    ( NSIZ ) , GHF    ( NSIZ ) , GFSH( NSIZ ) ,
     1          GFM    ( NSIZ )
      DIMENSION SO     ( NSIZ ) , TO     ( NSIZ )
      DIMENSION SRF    ( NSIZ )
      DIMENSION CRF    ( NSIZ ) , CRO    ( NSIZ )
      DIMENSION CPHI   ( NSIZ ) , CPHIO  ( NSIZ ) , CPHIA  ( NSIZ )
      DIMENSION NCODV  ( NSIZ ) , BCV    ( NSIZ ) , NOPPV  ( NSIZ ) ,
     1          MDFV   ( NSIZ )
      DIMENSION NCODT  ( NSIZ ) , BCT    ( NSIZ ) , NOPPT  ( NSIZ ) ,
     1          MDFT   ( NSIZ )
```

```
      DIMENSION NCODS ( NSIZ ) , BCS    ( NSIZ ) , NOPPS ( NSIZ ) ,
     1          MDFS  ( NSIZ )
      DIMENSION NCODF ( NSIZ ) , BCF    ( NSIZ ) , NOPPF ( NSIZ ) ,
     1          MDFF  ( NSIZ )
      DIMENSION NCODC ( NSIZ ) , BCC    ( NSIZ ) , NOPPC ( NSIZ ) ,
     1          MDFC  ( NSIZ )
      DIMENSION NCOD  ( NSIZ ) , BC     ( NSIZ ) , NOPP  ( NSIZ ) ,
     1          MDF   ( NSIZ )
      DIMENSION NDNV  ( NSIZ ) , NDNT   ( NSIZ ) , NDNS  ( NSIZ )
      DIMENSION NDNF  ( NSIZ ) , NDNC   ( NSIZ )
      DIMENSION NSDOF ( NSIZ , 4 ) , PPVL ( NSIZ )
      DIMENSION ISSB  ( NSIZ ) , NSSB   ( NSIZ , 3 ) , PBU (NSIZ)
      DIMENSION PSOUT ( NSIZ )    , NNEE ( NNOD )
      DIMENSION PMG   ( NSIZ )    , PRHS ( NSIZ )    , VHS ( NSIZ ,5 )
      DIMENSION VPROP ( 30   )    , NWR  ( 10   )
C
C....            FRONTAL ROUTINE  VARIABLES
C
      DIMENSION LDEST ( NSTF  ) ,
     1          LHED  ( MAXFR ) ,
     2          NK    ( NSTF  ) ,
     3          LPIV  ( MAXFR ) ,
     4          JMOD  ( MAXFR ) , QQ     ( MAXFR ) ,
     5          PVKOL ( MAXFR ) , R1     ( NSIZ  ) ,
     6          EQ    ( MAXFR   , MAXFR )
C
C....            GAUSSIAN QUADRATURE INTEGRATION VARIABLES
C
      DIMENSION GAUSS ( 7 , 7 ) , WT ( 7 , 7 )
C
C....            ELEMENT STIFFNESS, MASS , LOAD VECTOR MATRICES
C
      DIMENSION ELSTIF ( NSTF , NSTF ) , ELF ( NSTF ) , ELF1 ( NSTF )
      DIMENSION DMASS  ( NSTF , NSTF )
C.....................................................
C     GAUSSIAN QUADRATURE INTEGRATION DATA           .
C.....................................................
      DATA GAUSS/7*0.0D0,.57735027D0,-.57735027D0,5*0.0D0,-.77459667D0,
     *0.0D0,.77459667D0,4*0.0D0,-.8611363116D0,-.3399810435D0,
     *.3399810435D0,.8611363116D0,3*0.0D0,
     *-.90617985D0,-.53846931D0,0.0D0,.53846931D0,.90617985D0,2*0.0D0,
     *-.93246951D0,-.66120939D0,
     *-.23861918,.23861918D0,.66120939D0,.93246951D0,0.0D0,
     *-.94910791D0,-.74153119D0,-.40584515D0,0.0D0,.40584515D0,
     *.74153119D0,.94910791D0/
C      DATA GAUSS/-.96028986D0,-.79666648D0,-.52553241D0,-.18343464D0,
C     *.18343464D0,.52553241D0,.79666648D0,.96028986D0/

      DATA WT/2.0D0,6*.0D0,2*1.0,5*.0D0,.55555555D0,.88888888D0,
     *.55555555D0,4*0.0D0,
     *.3478548451D0,.6521451548D0,.6521451548D0,.3478548451D0,3*0.0D0,
     *.23692689D0,.47862867D0,.56888889,.47862867D0,.23692689D0,2*0.0D0,
     *.17132449D0,.36076157D0,.46791393D0,.46791393D0,.36076157D0,
     *.17132449D0,0.0D0,
     *.12948497D0,.27970539D0,.38183005D0,.41795918,.38183005D0,
     *.27970539D0,.12948497D0/
C       DATA WT/.10122854D0,.22238103D0,.31370665D0,.36268378D0,
C      *.36268378D0,.31370665D0,.22238103D0,.10122854D0/


C       DATA GAUSS/4*0.0D0,.57735027D0,-.57735027D0,2*0.0D0,-.77459667D0,
C      *0.0D0,.77459667D0,0.0D0,-.8611363116D0,-.3399810435D0,
C      *.3399810435D0,.8611363116D0/
C       DATA WT/2.0D0,3*0.0D0,2*1.0D0,2*0.0D0,.55555555D0,.88888888D0,
C      *.55555555D0,0.0D0,.3478548451D0,.6521451548D0,.6521451548D0,
C      *.3478548451D0/


C       DATA GAUSS/3*0.0D0,.57735027D0,-.57735027D0,0.0D0,-.77459667D0,
C      *0.0D0,.77459667D0/
C       DATA WT/2.0D0,2*0.0D0,2*1.0D0,0.0D0,.55555555D0,.88888888D0,
C      *.55555555D0/
C........................................
C        OPEN FILES FOR I/O                     .
C........................................
      OPEN( UNIT=14 ,FORM='UNFORMATTED' , STATUS='SCRATCH',RECL=MRCL )
      OPEN( UNIT=10 ,FILE='FERROR.LOG'   ,FORM='FORMATTED')
      OPEN( UNIT=12 ,FILE='OLDMESH',FORM='UNFORMATTED',STATUS='UNKNOWN')
      OPEN( UNIT=20 ,FILE='RESMSH.BIN',FORM='UNFORMATTED'
     1              ,STATUS='UNKNOWN')
      OPEN( UNIT=22 ,FILE='GEO.BIN',FORM='UNFORMATTED',STATUS='UNKNOWN')
      OPEN( UNIT=30 ,FILE='FREEOUT',FORM='UNFORMATTED')
      OPEN( UNIT=31 ,FILE='CARBOUT',FORM='UNFORMATTED')
      OPEN( UNIT=32 ,FILE='CARBPHI',FORM='UNFORMATTED')


C............................................................
C       DEFAULT VALUES FOR THE                                .
C       PROPERTIES OF VOID REGION AND                         .
C       COEEFICIENTS OF THE RELATION BETWEEN                  .
C       EFVF AND RELATIVE VISCOSITY USING EQUATION            .
C       R.V. = VPROP(21) + VPROP(22)*PHI                      .
.
C............................................................
      VPROP ( 8 ) =  0.1
      VPROP ( 9 ) =  1.1
      VPROP ( 10) =  0.027
      VPROP ( 11) =  1000.0
      PSLIM       =  1.0D+06
C..
      VPROP ( 21 ) = 1.00
      VPROP ( 22 ) = 5.500
C............................................
C     READING OF THE INPUT DATA  & PREPROCESSING  .
C............................................
      WRITE  ( * , 5000 )
      READ   ( * , 6000 ) FNAME1
C
      WRITE  ( * , 5003 )
      READ    ( * , 6000 ) FNAME2
C
      WRITE   ( * , 5006 )
      READ    ( * , 6005 ) NFREE
      IF ( NFREE.NE.1 ) THEN
         WRITE   ( * , 5004 )
         READ    ( * , 6005 ) NFDEF
         IF ( NFDEF.NE.1 ) THEN
            WRITE ( * , 5031 )
            READ  ( * , * ) VPROP (8)
            WRITE ( * , 5032 )
            READ  ( * , * ) VPROP (9)
            WRITE ( * , 5033 )
            READ  ( * , * ) VPROP (10)
            WRITE ( * , 5034 )
            READ  ( * , * ) VPROP (11)
            WRITE ( * , 5038 )
            READ  ( * , * ) PSLIM
         ENDIF
      ENDIF
C
      WRITE   ( * , 5007 )
      READ    ( * , 6005 ) NCARB
      IF ( NCARB.NE.1 ) THEN
         WRITE   ( * , 5008 )
         READ    ( * , 6005 ) NCDEF
         IF ( NCDEF.NE.1 ) THEN
            WRITE ( * , 5036 )
            READ  ( * , * ) VPROP (12)
            WRITE ( * , 5037 )
            READ  ( * , * ) VPROP (13)
         ENDIF
      ENDIF
C....
      WRITE   ( * , 5009 )
      READ    ( * , 6005 ) NCYL
      IF (NCYL.NE.1.AND.NCYL.NE.2.AND.NCYL.NE.3 ) NCYL=1
C....
      WRITE   ( * , 5010 )
      READ    ( * , 6005 ) NTRAN
      IF ( NTRAN .NE. 1 ) THEN
         WRITE   ( * , 5011 )
         READ    ( * , 6005 ) NITNS
         IF (NITNS .NE. 1 .AND. NITNS. NE. 2 .AND. NITNS.NE.3 ) NITNS=1
      ENDIF
C....
      WRITE   ( * , 5015 )
      READ    ( * , 6005 ) NNISO
C....
      IF (NNISO.NE.1) THEN
         WRITE   ( * , 5020 )
         READ    ( * , 6005 ) NSUPG
      ENDIF
C....
```

```
      IF (NTRAN .NE. 1 .AND. NTRAN. NE. 2 .AND. NTRAN.NE.3 ) NTRAN=1
      IF (NNISO .NE. 1 .AND. NNISO. NE. 2 ) NNISO=1
      IF (NSUPG .NE. 1 .AND. NSUPG. NE. 2 .AND. NSUPG.NE.3 ) NSUPG=1
C
C.... OPEN INPUT AND OUTPUT FILES (ASCII)
C
      OPEN ( UNIT=1 , FILE=FNAME1 , FORM='FORMATTED' )
      OPEN ( UNIT=2 , FILE=FNAME2 , FORM='FORMATTED' )
C....
      IF ( NFREE.NE.1.AND.NITNS.EQ.1 )  THEN
         WRITE  ( * , 5022 )
         READ   ( * , 6000 ) FNAME3
         OPEN ( UNIT=3 , FILE=FNAME3 , FORM='FORMATTED' )
      ENDIF
C
C.....DETERMINATION OF THE SYSTEM DATE
C     IDATE IS A SYSTEM SUBROUTINE OF LPI-FORTRAN
C
C     CALL IDATE  ( IMON,IDAY,IYR)
C.....
      PRINT *
      PRINT *,' READING INITIAL AND CONTROL DATA'
C     WRITE  ( 2 , 5024 )  IDAY,IMON,IYR
      READ   ( 1 , 6010 )  TITLE
      WRITE  ( 2 , 5025 )  TITLE
C .........................
      WRITE  ( 2 , 5026 )  NTRAN,NNISO,NSUPG,NCYL
C..........................
      READ   ( 1 , 6015 )  NEM, NNM, NPE, NBD1, NBF, NITER, IVIS, NSB
      READ   ( 1 , 6015 )  ( NWR (I) , I=1,3 )
      READ   ( 1 , 6020 )  VPROP(1) , VPROP(2)  , VPROP(15) , VPROP(14)
      READ   ( 1 , 6020 )  VPROP(3) , VPROP(4)  , CP     , AK
      READ   ( 1 , 6020 )  VPROP(5) , VPROP(6)  , VPROP(7)
      IF ( NCYL.EQ.2 )     READ   ( 1 , 6020 )  VPROP(12) , VPROP(13)
      WRITE  ( 2 , 5027 )  NEM,NNM
      WRITE  ( 2 , 5030 )  VPROP(1)    , VPROP(2)
      WRITE  ( 2 , 5035 )  VPROP(15) , VPROP(14)
      WRITE  ( 2 , 5040 )  VPROP(3)    , VPROP(4)   , CP     , AK
      WRITE  ( 2 , 5041 )  IVIS
      WRITE  ( 2 , 5042 )  VPROP(5) , VPROP(6)  , VPROP(7)
      IF ( NCYL.EQ.2 )     WRITE  ( 2 , 5043 )  VPROP(12) , VPROP(13)
C.........................................
C     CHECK THE NO. OF ELEMENTS AND NODES     .
C.........................................
      IF (NNM .GT. NNOD) THEN
         WRITE ( * , 5045 ) NNOD
         STOP
      ELSEIF (NEM .GT. NELM) THEN
         WRITE ( * , 5050 ) NELM
         STOP
      ENDIF
C.........................................
C     EVALUTIONS OF NDN , NOPP , MDF          .
C.........................................
      NDFV = 2
      NDFT = 1
      NDFS = 1
      NDFF = 1
      NDFC = 1
      CALL ARRZRI ( NDNV   , NSIZ )
      CALL ARRZRI ( NDNT   , NSIZ )
      CALL ARRZRI ( NDNS   , NSIZ )
      CALL ARRZRI ( NDNF   , NSIZ )
      CALL ARRZRI ( NDNC   , NSIZ )
      CALL ARRZRI ( MDFV   , NSIZ )
      CALL ARRZRI ( MDFT   , NSIZ )
      CALL ARRZRI ( MDFS   , NSIZ )
      CALL ARRZRI ( MDFF   , NSIZ )
      CALL ARRZRI ( MDFC   , NSIZ )
      CALL ARRZRI ( NOPPV , NSIZ )
      CALL ARRZRI ( NOPPT , NSIZ )
      CALL ARRZRI ( NOPPS , NSIZ )
      CALL ARRZRI ( NOPPF , NSIZ )
      CALL ARRZRI ( NOPPC , NSIZ )
C...
      DO I= 1,NELM
         NDNV (I) = NPE*NDFV
         NDNT (I) = NPE*NDFT
         NDNS (I) = NPE*NDFS
         NDNF (I) = NPE*NDFF
         NDNC (I) = NPE*NDFC
      ENDDO
      K=-1
      DO I= 1,NNM
         K=K+2
         MDFV  (I) = NDFV
         NOPPV (I) = K
         MDFT  (I) = NDFT
         NOPPT (I) = I
         MDFS  (I) = NDFS
         NOPPS (I) = I
         MDFF  (I) = NDFF
         NOPPF (I) = I
         MDFC  (I) = NDFC
         NOPPC (I) = I
      ENDDO
C.......................................
C     PRIMARY DEGREE OF FREEDOM         .
C.......................................
      CALL ARRZRI ( NCODV , NSIZ )
      CALL ARRZRI ( NCODT , NSIZ )
      CALL ARRZRI ( NCODS , NSIZ )
      CALL ARRZRI ( NCODF , NSIZ )
      CALL ARRZRI ( NCODC , NSIZ )
      CALL ARRZRI ( NCOD  , NSIZ )
      CALL ARRZRF ( BCV , NSIZ )
      CALL ARRZRF ( BCT , NSIZ )
      CALL ARRZRF ( BCS , NSIZ )
      CALL ARRZRF ( BCF , NSIZ )
      CALL ARRZRF ( BCC , NSIZ )
      CALL ARRZRF ( BC   , NSIZ )
      CALL ARRZRF ( PPVL , NSIZ )
C....
      IF ( NBD1  .NE.  0  ) THEN
C.....
C     IF YOU WANT TO PRINT THE B.C DATA IN OUTPUT FILE
C     REMOVE THE 'C' FROM THE WRITE COMMANDS
C.....
C         WRITE ( 2 , 5055 )
          DO 10 I= 1,NBD1
             READ    ( 1 , 6025 ) NODP,CU,CV,CT,CF,CC,
     1                            VAL1,VAL2,VAL3,VAL4,VAL5,PPVL(NODP)
C            WRITE   ( 2 , 5060 ) NODP,CU,CV,CT,CF,CC,
C    1                            VAL1,VAL2,VAL3,VAL4,VAL5,PPVL(NODP)
             IF (CU.EQ.'U') THEN
                NMDOF=NODP+NODP-1
                NCODV   ( NMDOF ) = 1
                BCV     ( NMDOF ) = VAL1
             ENDIF
             IF (CV.EQ.'V') THEN
                NMDOF = NODP+NODP
                NCODV ( NMDOF ) = 1
                BCV   ( NMDOF ) = VAL2
             ENDIF
C....................................
             IF (CU.EQ.'S') THEN
                NMDOF=NODP+NODP-1
                NCODV   ( NMDOF ) = 2
                BCV     ( NMDOF ) = VAL1
             ENDIF
             IF (CV.EQ.'P') THEN
                NMDOF = NODP+NODP
                NCODV ( NMDOF ) = 2
                BCV   ( NMDOF ) = VAL2
             ENDIF
C.................................
             IF (CT.EQ.'T') THEN
                NCODT ( NODP ) = 1
                BCT   ( NODP ) = VAL3
             ENDIF
C.................................
             IF (CF.EQ.'F') THEN
                NCODF ( NODP ) = 1
                BCF   ( NODP ) = VAL4
             ENDIF
C.................................
             IF ( CU   .EQ. 'U' .AND .CV   .EQ. 'V'  .AND.
     1            VAL1 .EQ. 0.0 .AND .VAL2 .EQ. 0.0  )     THEN
                  NCODS   ( NODP ) = 1
                  BCS     ( NODP ) = 0.0
             ENDIF
C.................................
             IF (CC.EQ.'C') THEN
                NCODC ( NODP ) = 1
                BCC   ( NODP ) = VAL5
```

```
              ENDIF
C.................................
 10         CONTINUE
C.....
      ENDIF
C............................................
C      SLIP LAYER DATA                        .
C............................................
 470  IF (NBF .EQ. 0 ) GOTO 400
      WRITE ( 2 , 5062 )
      DO 12 I=1,NBF
         READ  ( 1 , 6026 )  NSDOF(I,4),( NSDOF(I,J), J=1,3 )
         WRITE ( 2 , 5063 )  NSDOF(I,4),( NSDOF(I,J), J=1,3 )
 12   CONTINUE
C...................................................................
C     CALC. NO. EQUATIONS FOR FLOW , HEAT  EQ.  AND STREAM FUNCTION .
C...................................................................
 400  NEQ = NNM*2
      NET = NNM
      NES = NNM
      NEF = NNM
      NEC = NNM
      IF ( NEQ .GT. NSIZ ) THEN
         WRITE ( * , 5065 )
         STOP
      ENDIF
C.................................................
C     CHECK THE TYPE OF THE AIR COMPRESSIBILITY  .
C     APPLICATION
C.................................................
      IF ( NFREE.EQ.3 ) THEN
         DO N=1,NEF
            IF (NCODF(N).EQ.1) THEN
               NFREE =2
               GOTO 470
            ENDIF
         ENDDO
      ENDIF
C......................................................
C     NODAL POINTS DATA                               .
C                                                     .
C     IF YOU WANT TO ECHO-PRINT THE GEOMETRICAL DATA  .
C     IN OUTPUT FILE REMOVE THE 'C' FROM THE THREE    .
C     FOLLOWING WRITE COMMAND LINES                   .
C......................................................

      CALL ARRZRF ( X   , NNOD )
      CALL ARRZRF ( Y   , NNOD )
C..   WRITE ( 2 , 5070 )
      DO 14 I= 1,NNM
         READ  ( 1 , 6030 )     X(I) , Y(I)
C..      WRITE ( 2 , 5075 ) I , X(I) , Y(I)
 14   CONTINUE
C............................................
C     CONNECTIVITY DATA                      .
```

```
C........................................

      DO 16 I=1,NEM

         READ  ( 1 , 6035 ) J , ( NOD (J,K) , K= 1,NPE )
C..      WRITE ( 2 , 5085 ) J , ( NOD (J,K) , K= 1,NPE )


 16   CONTINUE
C..........................................................
C     CHANGE OF VELOCITY BOUNDARY CONDITIONS FOR CYLINDRICAL COORDINATE  .
C..........................................................
      IF ( NCYL.EQ.2 ) THEN
         DO I=1,NNM
            CALL CAR2CYL ( X(I) , Y(I) , RAD , THH )
            NDX = I+I-1
            NDY = I+I
            BXX = BCV (NDX)
            BYY = BCV (NDY)
            BCV (NDX) = BXX *   DCOS(THH)  +  BYY * DSIN(THH)
            BCV (NDY) = BXX * (-DSIN(THH)) +  BYY * DCOS(THH)
         ENDDO
      ENDIF
C................................................
C     FIND LAST APPEAREANCE OD EACH NODE (PRE-FRONT)  .
C................................................
      DO I= 1,NEM
         DO J=1,NPE
            NOP (I,J) = NOD (I,J)
         ENDDO
      ENDDO
      CALL   PREFNT  ( NNM , NEM , NOP , NPE , NELM )
C...........................................
C     INITIALIZING OF GLOBAL VECTORS            .
C...........................................
      CALL ARRZRF ( SO    , NSIZ )
      CALL ARRZRF ( GF    , NSIZ )
      CALL ARRZRF ( GFI   , NSIZ )
      CALL ARRZRF ( GFA   , NSIZ )
      CALL ARRZRF ( TO    , NSIZ )
      CALL ARRZRF ( TF    , NSIZ )
      CALL ARRZRF ( TFI   , NSIZ )
      CALL ARRZRF ( TFA   , NSIZ )
      CALL ARRZRF ( SRF   , NSIZ )
      CALL ARRZRF ( GFS   , NSIZ )
      CALL ARRZRF ( GFSI  , NSIZ )
      CALL ARRZRF ( GFSO  , NSIZ )
      CALL ARRZRF ( CRF   , NSIZ )
      CALL ARRZRF ( CRO   , NSIZ )
      CALL ARRZRF ( CPHI  , NSIZ )
      CALL ARRZRF ( CPHIO , NSIZ )
      CALL ARRZRF ( CPHIA , NSIZ )
      CALL ARRZRF ( PRHS  , NSIZ )
      CALL ARRZRF ( GFM   , NSIZ )
C...........................................
```

```
      DO I= 1,NSIZ
         DO J=1,5
            VHS ( I,J ) = 0.0
         ENDDO
      ENDDO
C.........................................
C     READ INITIAL AND TRANSIENT DATA           .
C.........................................
C
C..   INITIAL VELOCITY, TEMPERATURE AND CONCENTRATION
C
      READ  ( 1 , 6040 ) VINIT,TINIT,CINIT
C
C..   TOLERANCE
C
      READ  ( 1 , 6045 ) TOLE1,TOLE2,TOLE3
      WRITE ( 2 , 5088 ) TOLE1,TOLE2
C
C..         TRANSIENT DATA
C
      READ  ( 1  ,6050 ) STIME,DT,NTIME,THETA
      IF ( NTRAN .EQ. 2 .OR. NTRAN.EQ.3 ) THEN
        WRITE ( 2 , 5090 ) STIME,DT,NTIME,THETA,VINIT,TINIT,CINIT
        IF ( NITNS .EQ. 1 ) THEN
           OPEN (UNIT=4,FILE='TRANS.OUT',FORM='FORMATTED')
           WRITE ( 4 , 5095 )  (NWR(I),I=1,3),FNAME1,NSUPG
        ELSE
         OPEN (UNIT=4,FILE='TRANS.OUT',FORM='FORMATTED',ACCESS='APPEND')
         WRITE ( 4 , 5096 )  STIME
         ENDIF
      ENDIF
C.....................................
C   DEFINITION OF THE INITIAL CONDITIONS   .
C.....................................
      IF ( NITNS .EQ. 1 .OR.  NTRAN .EQ.1 ) THEN
         DO I= 1,NEQ
            SO(I)=VINIT
         ENDDO
         DO I= 1,NET
            TO(I)=TINIT
         ENDDO
         IF ( NFREE.NE.1 ) THEN
            DO I=1,NEF
               READ ( 3 , 6041 ) K,GFSO(K)
            ENDDO
         ENDIF
         DO I=1,NEC
            CRO(I)     =  CINIT
            CPHIO(I)   =  CINIT
         ENDDO
      ELSE
         REWIND 20
         DO I= 1,NNM
               READ (20) SO(I+I-1)     , SO(I+I) , TO(I)     ,
     1                   GFSO(I)       , CRO(I)  , CPHIO(I) ,
     2                   CORP ( I,1 ) , CORP (I,2)
         ENDDO
         IF (NITNS.EQ.3) THEN
C....       CALCULATION OF THE MESH VELOCITY
            DO I=1,NNM
               GFM (I+I-1) = ( X(I)-CORP (I,1) ) /DT
               GFM (I+I  ) = ( Y(I)-CORP (I,2) ) /DT
            ENDDO
         ENDIF
C
C        FINDING THE INITIAL DATA FOR ALE METHOD USING
C        INTERPOLATION TECHNIQUE
C
C        CALL   MSHINI ( X    , Y   , NNM  , NNOD  , NELM  , NPE    ,
C    1                   NSIZ , CORP, NODPV, AINIT , SO    , TO     ,
C    2                   GFSO , CRO , CPHIO, NCARB , NFREE , NNISO  ,
C    3                   DT                                         )
      ENDIF
C.............................................................
C     SETUP TIME VARIABLES FOR STEADY SOLUTION                .
C.............................................................
      IF (  NTRAN.EQ.1 ) THEN
         DT=1.0
         NTIME=1
      ENDIF
C.............................................................
C DATA FOR BOUNDARY INTEGRAL IN STREAM FUNCTION FORMULATION .
C.............................................................
      CALL ARRZRI ( ISSB , NSIZ )
      IF  ( NSB.EQ.0 ) GOTO 450
      DO I= 1,NSB
         READ  ( 1 , 6055 )  ISSB(I),( NSSB (I,J), J=1,3 ) , PBU(I)
C.....    WRITE ( 2 , 5063 )  ISSB(I),( NSSB (I,J), J=1,3 ) , PBU(I)
      ENDDO
C...............................................
450   CLOSE (1)
C...............................................
C  DETERMINATION OF REDUCED & FULL INTG. INDEX   .
C...............................................
      IF (NPE.EQ.4) THEN
            IR=1
            IF=7
      ELSEIF (NPE.EQ.8.OR.NPE.EQ.9) THEN
            IR=2
            IF=3
      ENDIF

C     IF (IF.EQ.8) THEN
C     WT(1,1)= 10122854D0
C     WT(2,2)=.22238103D0
C     WT(3,3)=.31370665D0
C     WT(4,4)=.36268378D0
C     WT(5,5)=.36268378D0
C     WT(6,6)=.31370665D0
C     WT(7,7)=.22238103D0
```

```
C      WT(8,8)=.10122854D0
C       GAUSS(1,1)=-.96028986D0
C      GAUSS(2,2)=-.79666648D0
C      GAUSS(3,3)=-.52553241D0
C      GAUSS(4,4)=-.18343464D0
C      GAUSS(5,5)= .18343464D0
C      GAUSS(6,6)= .52553241D0
C      GAUSS(7,7)= .79666648D0
C      GAUSS(8,8)= .96028986D0
C       END IF

C.............................................
C  FIND CONNECTIVITY OF EACH NODE TO ADJ. ELE. .
C.............................................
      CALL ANODAE ( NNM , NEM , NPE , NOD , NNEE , NNOD , NELM )
C.............................................
C  FIND GLOBAL LUMPED MASS MATRIX FOR V.R.      .
C.............................................

      CALL PVRGMX ( NNM    , NEM , NPE , NOD , NNOD , NELM  , PMG   , IR,
     1              GAUSS , WT  , X   , Y   , NSTF , DMASS , IF , NSIZ)

C..............................................
C     START OF THE TIME STEPPING LOOP          .
C..............................................
C
      DTI = DT
      IF ( NITNS.EQ.0 ) THEN
         TIME = 0.0
      ELSE
         TIME = STIME
      ENDIF
C....................................................
      PRINT *,' START OF FINITE ELEMENT CALCULATION    '
C....................................................
      DO 60 NT= 1,NTIME
C..............................................
      IDIV  = 0
      TIMEI = TIME
 435  IEND  = 0
      DT = DTI / DFLOAT ( IDIV+1 )
      TIME=TIMEI+DT
C..............................................
C     INITIAL ESTIMATE  ASSIGNMENT             .
C..............................................
      CALL RSAVE  ( GFI   , SO    , NEQ )
      CALL RSAVE  ( TFI   , TO    , NET )
      CALL RSAVE  ( GFSI  , GFSO  , NEF )
      CALL RSAVE  ( CPHIA , CPHIO , NEC )
C...
      DO 1000 ITER = 1,NITER
C.............................................
      CALL RSAVE ( GFA , GFI, NEQ )
      CALL RSAVE ( TFA , TFI, NET )
      IF ( NFREE.EQ.1 )  THEN
```

```
          DO K=1,NEF
             GFS  (K) = 1.0
             GFSI (K) = 1.0
             GFSO (K) = 1.0
          ENDDO
      ENDIF
C.........................................
C     MESH VELOCITY IS SET TO FLUID VELOCITY   .
C     IN  PURE LAGRANGIAN ANALYSIS             .
C.........................................
C     CALL RSAVE ( GFM , GFI, NSIZ )
C
C
      IF ( NCYL.NE.2 ) THEN
C.........................................
C     CALCULATION OF VELOCITIES                .
C.........................................
      CALL FLOWCN (
     1 GF    , GFI   , TFI   , SO    , GFSI  , GFSO , TO              ,
     2 GAUSS , WT    , CPHI  , GFM                                    ,
     3 X     , Y     , NOD   , NOP                                    ,
     4 BCV   , NCODV , NOPPV , MDFV  , NDNV                           ,
     5 NPE   , IR    , IF    , DT    , THETA , NDFV , NEM             ,
     6 NEQ   , NTRAN , NCARB , NCYL                                   ,
     7 PRHS  , VPROP , IVIS                                           ,
     8 NSIZ  , NSTF  , NELM  , NNOD  , MAXFR                          ,
     9 R1    , ELF   , ELSTIF , DMASS , VHS                           ,
     1 LDEST , NK    , EQ    , LHED  , LPIV                           ,
     2 JMOD  , QQ    , PVKOL , NSB   , ISSB  , NSSB , PBU             ,
     3 NCOD  , BC    , NOPP  , MDF   , NSDOF , PPVL , NBF             )
C....
      ELSE
C.......................................................
C    CALCULATION OF VELOCITIES ( CYLINDRICAL AND CEF EQUATION  ) .
C.......................................................
C
C..   TRANSFORMATION OF NODAL POINT COORDINATES AND VELOCITY VECTOR  .
C..   COMPONENTS TO CYLINDRICAL COORDINATE SYSTEM
C
      DO I=1,NNM
         CALL CAR2CYL ( X(I) , Y(I) , RAD , THH )
         X(I)=RAD
         Y(I)=THH
         CALL VCA2CL ( GF (I+I-1)   ,  GF (I+I)   , THH )
         CALL VCA2CL ( GFI(I+I-1)   ,  GFI(I+I)   , THH )
         CALL VCA2CL ( SO (I+I-1)   ,  SO (I+I)   , THH )
      ENDDO
C...............................................................
      CALL  FLCYL  (
     1 GF    , GFI   , TFI   , SO    , GFSI  , GFSO , TO              ,
     2 GAUSS , WT    , CPHI                                           ,
     3 X     , Y     , NOD   , NOP                                    ,
     4 BCV   , NCODV , NOPPV , MDFV  , NDNV                           ,
     5 NPE   , IR    , IF    , DT    , THETA , NDFV , NEM             ,
     6 NEQ   , NNM   , NTRAN , NCARB                                  ,
     7  VPROP , IVIS                                                   ,
     8  NSIZ  , NSTF   , NELM    , NNOD   , MAXFR                       ,
     9  R1    , ELF    , ELSTIF  , DMASS  , VHS                         ,
     1  LDEST , NK     , EQ      , LHED   , LPIV                        ,
     2  JMOD  , QQ     , PVKOL                                          ,
     3  NCOD  , BC     , NOPP    , MDF    , NSDOF , PPVL , NBF          )
C
C..    PRESSURE CALCULATION
C
       CALL ARRZRF ( PRHS   , NSIZ )
       CALL PRSCYL ( NEM    , GAUSS , NPE   , GF     , NOD           ,
     1               X      , Y     , IR    , IF     , NELM , NNOD   ,
     2               NSIZ   , VPROP , TFI   , CPHI   , NCARB, IVIS   ,
     3               PRHS   , WT    , GFSI  , PMG    , NNM           )
C
C..    TRANSFORMATION OF NODAL POINT COORDINATES AND VELOCITY VECTOR
C..    COMPONENTS TO CARTESIAN COORDINATE SYSTEM
C
       DO I=1,NNM
          CALL CYL2CAR ( XC    , YC    , X(I) , Y(I) )
          THH = Y(I)
          X(I)=XC
          Y(I)=YC
          CALL VCL2CA  ( GF  (I+I-1)   ,   GF  (I+I)   , THH )
          CALL VCL2CA  ( GFI(I+I-1)    ,   GFI(I+I)    , THH )
          CALL VCL2CA  ( SO  (I+I-1)   ,   SO  (I+I)   , THH )
       ENDDO
       ENDIF
C.............................................
C         CALCULATION OF ERROR NORM          .
C.............................................
       CALL VNORM (VN , NEQ , GFA    , GF    , NSIZ )
       WRITE ( * , 5110 ) NT,ITER,VN
       WRITE (10 , 5110 ) NT,ITER,VN
C..................................................
C      CALCULATION OF VELOCITY FIELD AT HALF-TIME  .
C      STEP ( ONLY FOR TAYLOR-GALERKIN SCHEME )    .
C..................................................
       IF ( NTRAN.EQ.3 ) THEN
          DO I=1,NEQ
             GHF (I)= ( GF(I)+SO(I))/2.0
             GFSH(I)= ( GFS(I)+GFSO(I))/2.0
          ENDDO
       ENDIF
C...............................................
C      CULCULATION OF VELOCITY GRADIENT        .
C      COMPONENTS USING VARIATIONAL RECOVERY   .
C      FORMULATION                             .
C...............................................
       CALL         VISRHD ( VHS    , GF    , X     , Y     , NOD  , IR   ,
     1                       IF     , GAUSS , WT    , NPE   , NELM , NNOD ,
     2                       NSIZ   , NSTF  , PMG   , NEM   , NNM         )
C...............................................
C     CHECK FOR ISOTHERMAL CASE                .
C...............................................
```

```
      IF ( NNISO .EQ. 1 ) THEN
            CALL RSAVE ( TF   , TFI, NET )
            GOTO 420
      ENDIF
C................................................
      IF ( NTRAN.EQ.1 .OR. NTRAN.EQ.2  ) THEN
C................................................
C     CALCULATION OF TEMPERATURE                 .
C................................................
      CALL TMPRUR (
     1 TF     , TFI   , GF     , TO    , SO                               ,
     2 GAUSS  , WT    , VHS    , GFSI  , GFSO   , GFM                     ,
     3 X      , Y     , NOD    , NOP   , CPHI                             ,
     4 BCT    , NCODT , NOPPT  , MDFT  , NDNT                             ,
     5 NPE    , IR    , IF     , DT    , THETA  , NDFT  , NEM             ,
     6 NET    , NNM   , NTRAN  , NSUPG , NCARB                            ,
     7 AK     , CP    , VPROP                                             ,
     8 NSIZ   , NSTF  , NELM   , NNOD  , MAXFR  , IVIS                    ,
     9 R1     , ELF   , ELSTIF , DMASS , ELF1                             ,
     1 LDEST  , NK    , EQ     , LHED  , LPIV                             ,
     2 JMOD   , QQ    , PVKOL  , NCYL                                     ,
     3 NCOD   , BC    , NOPP   , MDF                                      )
C.....................................................................
      ELSE
C.....................................................................
C     SOLUTION OF ENERGY EQUATION USING TAYLOR-GALERKIN              .
C.....................................................................
      CALL TMPTGL (
     1 TF     , TFI   , GF     , TO    , SO     , THF                     ,
     2 GAUSS  , WT    , VHS    , GHF   , GFSO   , GFSH                    ,
     3 X      , Y     , NOD    , NOP   , CPHI                             ,
     4 BCT    , NCODT , NOPPT  , MDFT  , NDNT                             ,
     5 NPE    , IF    , DT     , NDFT  , NEM                              ,
     6 NET    , NNM   , NTRAN  , NSUPG , NCARB                            ,
     7 AK     , CP    , VPROP                                             ,
     8 NSIZ   , NSTF  , NELM   , NNOD  , MAXFR  , IVIS                    ,
     9 R1     , ELF   , ELSTIF , DMASS , ELF1                             ,
     1 LDEST  , NK    , EQ     , LHED  , LPIV                             ,
     2 JMOD   , QQ    , PVKOL                                             ,
     3 NCOD   , BC    , NOPP   , MDF                                      )
C................................................
      ENDIF
      CALL VNORM (TN , NET , TFA   , TF    , NSIZ )
      WRITE ( * , 5115 ) TN
      WRITE (10 , 5115 ) TN
C.....................................................
C     CALCULATION OF THE VALUE OF F                   .
C.....................................................
 420  IF ( NFREE.NE.1) THEN
      CALL FRSFVL (
     1 GFS    , GFSI  , GFSO   , GF    , GFM                              ,
     2 GAUSS  , WT                                                        ,
     3 X      , Y     , NOD    , NOP                                      ,
     4 BCF    , NCODF , NOPPF  , MDFF  , NDNF                             ,
     5 NPE    , IR    , IF     , DT    , THETA  , NDFF  , NEM             ,
     6  NEF    , NNM                                                    ,
     8  NSIZ   , NSTF   , NELM    , NNOD    , MAXFR , IVIS             ,
     9  R1     , ELF    , ELSTIF  , DMASS                               ,
     1  LDEST  , NK     , EQ      , LHED    , LPIV                      ,
     2  JMOD   , QQ     , PVKOL                                         ,
     3  NCOD   , BC     , NOPP    , MDF                                 )
C...
      CALL VNORM (FN , NEF , GFSI   , GFS   , NSIZ )
      WRITE ( * , 5120 ) FN
      WRITE (10 , 5120 ) FN
      ENDIF
C............................................
C  CONCENTRATION   CALCULATIONS               .
C............................................
      IF ( NCARB.NE.1) THEN
      CALL CARBON (
     1  CRF    , GF     , CRO     , GFS     , GFM                       ,
     2  GAUSS  , WT                                                     ,
     3  X      , Y      , NOD     , NOP                                 ,
     4  BCC    , NCODC  , NOPPC   , MDFC    , NDNC                      ,
     5  NPE    , IR     , IF      , DT      , THETA , NDFC , NEM        ,
     6  NEC    , NNM                                                    ,
     7  NSIZ   , NSTF   , NELM    , NNOD    , MAXFR                     ,
     8  R1     , ELF    , ELSTIF  , DMASS                               ,
     9  LDEST  , NK     , EQ      , LHED    , LPIV                      ,
     1  JMOD   , QQ     , PVKOL                                         ,
     2  NCOD   , BC     , NOPP    , MDF                                 )
C...................................................................
      CALL EFVF    (
     1  CPHI   , GF     , CPHIO   , CPHIA , CRF    , CRO , GFM          ,
     2  GAUSS  , WT     , VPROP   , GFS                                 ,
     3  X      , Y      , NOD     , NOP                                 ,
     4  BCC    , NCODC  , NOPPC   , MDFC    , NDNC                      ,
     5  NPE    , IR     , IF      , DT      , THETA , NDFC , NEM        ,
     6  NEC    , NNM                                                    ,
     7  NSIZ   , NSTF   , NELM    , NNOD    , MAXFR                     ,
     8  R1     , ELF    , ELSTIF  , DMASS                               ,
     9  LDEST  , NK     , EQ      , LHED    , LPIV                      ,
     1  JMOD   , QQ     , PVKOL                                         ,
     2  NCOD   , BC     , NOPP    , MDF                                 )
C...
      CALL VNORM (CN , NEC , CPHIA , CPHI , NSIZ )
      WRITE ( * , 5125 ) CN
      WRITE (10 , 5125 ) CN
      ENDIF
      WRITE ( * , '(1H )')
      WRITE (10 , '(1H )')
C............................................
C  STREAM FUNCTION CALCULATION                .
C............................................
C     CALL STRMFC (
C    1  SRF    , GF                                                     ,
C    2  GAUSS  , WT     , NSB     , ISSB    , NSSB                      ,
C    3  X      , Y      , NOD     , NOP                                 ,
C    4  BCS    , NCODS  , NOPPS   , MDFS    , NDNS                      ,
C     5  NPE    , IR     , IF                          , NDFS  , NEM               ,
C     6  NES    , NNM                                                              ,
C     7  NSIZ   , NSTF   , NELM    , NNOD   , MAXFR                                ,
C     8  R1     , ELF    , ELSTIF                                                  ,
C     9  LDEST  , NK     , EQ      , LHED   , LPIV                                 ,
C     1  JMOD   , QQ     , PVKOL                                                   ,
C     2  NCOD   , BC     , NOPP    , MDF                                           )
C.........................................
C     CALCULATION OF PRESSURE ( V.R. )          .
C.........................................
      IF ( NCYL.NE.2 ) THEN
        CALL        PRESS ( NEM    , GAUSS , NPE   , GF     , NOD   , NCYL ,
     1                      X      , Y     , IR    , IF     , NELM  , NNOD ,
     2                      NSIZ   , VPROP , TF    , VHS    , CPHI  , NCARB,
     3                      PSOUT  , NNM   , IVIS  , PRHS   , WT    , GFS  ,
     4                      PMG                                            )
C.......
C       CALCULATION OF PRESSURE FOR DISCRETE PENALTY METHOD
C......
C       CALL       PRESD  ( NE     , NPE , GAUSS ,WT , CPHI     ,
C     1                      VPROP , GF  , TF    , X , Y        , NCARB  ,
C     2                      NOD   , IR  , IF    , NELM , NNOD  , NEM    ,
C     3                      NSIZ  , NSTF, IVIS  , VHS  , PSOUT , GFS    ,
      ENDIF
C........................................
C     CHECK THE CONVERGENCE                   .
C........................................
      CALL RENWAL ( GFI   , GF   , NEQ , 1.00D00 )
      CALL RENWAL ( TFI   , TF   , NET , 1.00D00 )
      CALL RENWAL ( GFSI  , GFS  , NEF , 1.00D00 )
      CALL RENWAL ( CPHIA, CPHI, NEC , 1.00D00 )
      IF ( VN .LT. TOLE1 .AND.
     1     TN .LT. TOLE2 .AND.
     2     FN .LT. TOLE1 .AND.
     3     CN .LT. TOLE1      ) IEND=1
      IF ( ITER.EQ.NITER.AND.NTRAN.EQ.1   ) IEND = 1
      IF ( ITER.EQ.NITER.AND.NTRAN.NE.1.AND.IEND.NE.1   ) THEN
         IDIV = IDIV + 1
C.....   GOTO 435
         IEND = 1
      ENDIF
C...............................................
C     APPLICATION OF THE AIR COMPRESSIBILITY          .
C     BY THE MODIFICATION OF THE BOUNDARY CONDITIONS.
C     IN FREE SURFACE EQUATION                        .
C...............................................
      IF ( NFREE.EQ.3.AND.IEND.EQ.1 ) THEN
         DO I= 1,NEF
            IF ( DABS (PRHS(I)).GT.PSLIM.AND.GFS(I).LT.0.3 ) THEN
               NCODF (I) = 1
               BCF   (I) = 1.0
               IEND = 0
            ENDIF
         ENDDO
      ENDIF
```

```
C..........................................
C     OUTPUTING THE NODAL VALUES               .
C..........................................
      CALL       OUTPUT ( NNM   , GF  , TF  , NSIZ  , PSOUT  , NNEE  , PMG ,
     1                    CRF   , PRHS      , TIME  , NWR    , NTRAN       ,
     2                    ITER  , IEND      , VN    , TN     , GFS         ,
     3                    NFREE, NCARB      , VHS   , CPHI   , VPROP       )
C...........................................................................
      IF ( IEND.EQ.1 ) GOTO 416
C...........................................................................
 1000 CONTINUE
C..........................................
 416  CALL RSAVE ( SO , GF , NEQ )
      IF (NNISO.EQ.2)  CALL RSAVE ( TO , TF , NET )
      CALL RSAVE ( GFSO , GFS , NEF )
      CALL RSAVE ( CRO  , CRF , NEC )
      CALL RSAVE ( CPHIO, CPHI, NEC )
C..............................................
C                                                   .
C     UPDATE THE MESH COORDINATES BY                .
C     DISTANCE = VELOCITY * TIME                    .
C     ONLY IN THE CASE OF PURE LAGRANGIAN FORMULATION.
C                                                   .
C..............................................
C
C     CALL MSHUPD ( X , Y , GF , NNOD , NSIZ , DT , NNM )
C
C..............................
   60 CONTINUE
C..............................
C
C..
C     WRITING THE RESULTS OF THE LAST TIME STEP AND NODAL COORDINATES
C     IN A FILE (RESMSH.BIN)
C..
      REWIND 20
      DO N=1,NNM
         WRITE (20) SO(N+N-1), SO(N+N)  , TO(N)     ,
     1              GFSO(N)   , CRO(N)   , CPHIO(N) ,
     2              X(N)      , Y(N)
      ENDDO
C
C..
C     WRITING THE RESULTS OF THE LAST TIME STEP ( ONLY VELOCITY,
C     TEMPERATURE AND PRESSURE IN FILE GEO.BIN FOR USING IN GEOSTAR
C     ENVIROMENT
C..
      REWIND 22
      DO N=1,NNM
         VRES= DSQRT (GF(N+N-1)**2+GF(N+N)**2)
         WRITE (22) GF(N+N-1), GF(N+N)  , VRES , TF(N)   , PRHS(N)
      ENDDO
C
C...  WRITING THE NUMBER OF NODES AND ELEMENTS AND
C..   ELEMENT CONNECTIVITY DATA FOR INTERPOLATION METHOD
C
      REWIND 12
      WRITE ( 12 ) NNM,NEM
      DO I=1,NEM
         WRITE ( 12 ) (NOD(I,J),J=1,NPE )
      ENDDO
C
      ENDFILE 12
      ENDFILE 20
      ENDFILE 22
C................................
      ENDFILE 30
      ENDFILE 31
      ENDFILE 32
C................................
      CLOSE (2 )
      CLOSE (3 )
      CLOSE (10)
      CLOSE (12)
      CLOSE (20)
      CLOSE (30)
      CLOSE (31)
      CLOSE (32)
C................................................
C   FORMAT                                       .
C................................................
C
C.....WRITE FORMATS
C
 5000 FORMAT (1X,'ENTER INPUT  FILE NAME ___   ',$)
 5003 FORMAT (1X,'ENTER OUTPUT FILE NAME ___   ',$)
 5004 FORMAT (1X,'DEFAULT VALUES FOR VOID AREA PROPERTIES',/,
     1        1X,'AND PRESSURE LIMIT ?  1:YES , 2:NO  ',$)
 5005 FORMAT (1X,'BINARY FILE GENERATION FOR NODES AND ELEMENTS 1:NO , 2
     1:YES  ',$)
 5006 FORMAT (1X,'FREE SURFACE ANALYSIS (?)',/,
     1           ' 1: NO' ,/,
     2           ' 2: YES ( WITHOUT AIR COMPRESSIBILITY )',/,
     3           ' 3: YES ( WITH AIR COMPRESSIBILITY )')
 5007 FORMAT (1X,'CONCENTRATION CALC.   (?)  1:NO , 2:YES  ',$)
 5008 FORMAT (1X,'DEFAULT VALUES FOR R.V. EQUATION ?  1:YES , 2:NO  ',$)
 5009 FORMAT (1X,'TYPE OF RHEOLOGICAL EQUATION AND COORDINATE SYSTEM:',
     1      /,2X,' 1) GENERALIZED NEWTONIAN ( FLOW EQ. IN CARTESIAN)',
     2      /,2X,' 2) CEF ( FLOW EQ. IN CYLINDRICAL)',
     3      /,2X,' 3) GENERALIZED NEWTONIAN IN AXISYMMETRIC (R,Z) ')
 5010 FORMAT (1X,'TYPE OF TIME ANALYSIS:',
     1        /,2X,'1)  STEADY STATE ( INDEPENDENT OF TIME)',
     2        /,2X,'2)  TRANSIENT ( BOTH EQ. WITH THETA METHOD )',
     3        /,2X,'3)  TRANSIENT ( FLOW EQ. WITH THETA AND HEAT EQ. WIT
     4H TAYLOR-GALERKIN)')
 5011 FORMAT (3X  ,' INITIAL DATA OPTIONS :',
     1      /,6X,'1) INITIAL FROM INIPUT FILE ',
     2      /,6X,'2) PREVIOUSLY EXECUTED MODEL (EULERIAN)',
     3      /,6X,'3) PREVIOUSLY EXECUTED MODEL (ALE)')
 5015 FORMAT (1X,'TYPE OF TEMPERATURE ANALYSIS:',
    1         /,2X,'1) ISOTHERMAL ',
    2         /,2X,'2) NONISOTHERMAL')
 5020 FORMAT (1X,'TYPE OF UPWINDING TECHNIQUE FOR HEAT EQUATION',
    1         /,2X,'1) STANDARD GALERKIN METHOD',
    2         /,2X,'2) STREAMLINE UPWINDING          ',
    3         /,2X,'3) STREAMLINE UPWINDING / PETROV-GALERKIN')
 5022 FORMAT (1X,'ENTER INITIAL FILE FOR FREE SURFACE ___  ',$)
 5024 FORMAT (1X,/,' DATE  : ',I2,'/',I2,'/',I4)
 5025 FORMAT (1X,'TITLE : ',A,/)
 5026 FORMAT (1X,'A N A L Y S I S   T Y P E :',/,
    1         1X,'NTRAN = ',I4,/,
    2         1X,' 1 -  STEADY STATE ( INDEPENDENT OF TIME)',/,
    3         1X,' 2 -  TRANSIENT ( THETA METHOD )',/,
    4         1X,' 3 -  TRANSIENT ( THETA METHOD & TAYLOR-GALERKIN)',/,
    5         1X,'NNISO = ',I4,/,
    6         1X,' 1 -  ISOTHERMAL ',/,
    7         1X,' 2 -  NONISOTHERMAL ',/,
    8         1X,'NSUPG = ',I4,/,
    9         1X,' 1 -  STANDARD GALERKIN METHOD',/,
    1         1X,' 2 -  STREAMLINE UPWINDING',/,
    2         1X,' 3 -  STREAMLINE UPWINDING / PETROV-GALERKIN',/,
    3         1X,'NCYL = ',I4,/,
    4         1X,' 1 -  GENERALIZED NEWTONIAN ',/,
    5         1X,' 2 -  CEF                   ',/,
    6         1X,' 3 -  AXISYMMETRIC (GN)     ',/)
 5027 FORMAT (1X,'N O. O F   E L E M E N T S = ',I5,/,
    1         1X,'N O. O F   N O D E S       = ',I5,/)
 5030 FORMAT (1X,'POWER LAW CONSTANT...& POWER LAW INDEX..=',2D15.5)
 5031 FORMAT (1X,'VISCOSITY OF THE VOID AREA => ',$)
 5032 FORMAT (1X,'DENSITY   OF THE VOID AREA => ',$)
 5033 FORMAT (1X,'TH. COND. OF THE VOID AREA => ',$)
 5034 FORMAT (1X,'HEAT CAP. OF THE VOID AREA => ',$)
 5038 FORMAT (1X,'PRESSURE LIMIT             => ',$)
 5035 FORMAT (1X,'PENALTY.............=',D15.5,/,' DENSITY............
    1.=',D15.5  )
 5036 FORMAT (1X,'FIRST  COEFFICIENT OF R.V. EQ. => ',$)
 5037 FORMAT (1X,'SECOND COEFFICIENT OF R.V. EQ. => ',$)
 5040 FORMAT (1X,'TEMPERATURE SENS. ...=',F15.5,/,' REFERENCE TEMP. ....
    1.=',F15.5,/,' HEAT CAPACITY.......=',F15.5,/,' THERMAL COND. ....
    2...=',F15.5,/)
 5041 FORMAT (1X,'VISCOSITY EQUATION TYPE .....=',I5,/,
    1         7X,'1 : POWER LAW  *  2 : CARREAU MODEL',/)
 5042 FORMAT (1X,'RELAXATION TIME...............=',D15.5,/
    1         1X,'SLIP COEFFICIENT..............=',D15.5,/
    2         1X,'KINETICS CHARACTERISTIC TIME...=',D15.5,/)
 5043 FORMAT (1X,'FIRST  CHARAC. CONS. FOR PSIC...=',D15.5,/
    1         1X,'SECOND CHARAC. CONS. FOR PSIC...=',D15.5,/)
 5045 FORMAT (1X,/,'--ERROR-- MAXIMUM NO. OF NODES ',I7)
 5050 FORMAT (1X,/,'--ERROR-- MAXIMUM NO. OF ELEM. ',I7)
 5055 FORMAT (1X,'PRIMARY DEGREE OF FREEDOM',/)
 5060 FORMAT (1X,I5,2X,5A2,5F10.4,F10.6)
 5062 FORMAT (1X,/,'SLIP LAYER(S) DATA',/)
 5063 FORMAT (1X,4I5,F20.10)
 5065 FORMAT (1X,/,'--ERROR IN NO. OF INITIAL DEFINITION OF EQUATIONS')
 5070 FORMAT (1X,' NODAL POINTS COORDINATES (X,Y)  :  ',/)
 5075 FORMAT (5X,I5,2F10.4)
 5080 FORMAT (1X,/,'CONNEVTIVITY MATRIX :',/)
 5085 FORMAT (1X,10I5)
 5088 FORMAT (1X,' FLOW EQUATION TOLERANCE =', D15.5,/,
     1          1X,' TEMPERATURE EQ. TOLE    =', D15.5    )
 5090 FORMAT (1X,' STARTING TIME ....... = ', D15.5 , / ,
     1          1X,' TIME INCREMENT....... = ', D15.5 , / ,
     2          1X,' NO OF TIME.STEPS..... = ', I5      , / ,
     3          1X,' THETA................ = ', F10.4 , / ,
     4          1X,' INITIAL VELOCITY..... = ', F10.4 , / ,
     5          1X,' INITIAL TEMPERATURE.. = ', F10.4 , / ,
     6          1X,' INITIAL CONCENTRATION = ', F10.4      )
 5095 FORMAT (2X,'TRANSIENT OUTPUT RESULTS FOR NODES =',3I7,/,
     1         2X,'INPUT FILE NAME - - - ',A,/,
     2         2X,'NSUPG= ',I3)
 5096 FORMAT (2X,'RESTART FROM TIME = ',D15.5)
 5110 FORMAT (1X,'NT=',I5,' IT=',I2,' >E(FL)=',F7.4,$)
 5115 FORMAT (' >E(TP)=',F7.4,$)
 5120 FORMAT (' >E(FS)=',F7.4,$)
 5125 FORMAT (' >E(CR)=',F7.4,$)
C
C... READ FORMATS ...................
C
 6000 FORMAT (A)
 6005 FORMAT ( I3 )
 6010 FORMAT (A)
 6015 FORMAT (16I5)
 6020 FORMAT (4D15.5)
 6025 FORMAT (I5,2X,5A2,5F10.4,F10.6)
 6026 FORMAT (4I5,F20.10)
 6030 FORMAT (5X,2G20.8)
 6035 FORMAT (10I5)
 6040 FORMAT (3F10.4)
 6041 FORMAT (I5,F5.1)
 6045 FORMAT (3F20.9)
 6050 FORMAT (2F15.8,I5,F10.4)
 6055 FORMAT (4I5,D15.5)
C
C .........................................
C
      STOP
      END
C..............................................................
C     END OF THE MAIN PROGRAM                                 .
C..............................................................
C
C.........................................
C    SOLUTION OF FLOW EQUATIONS              .
C.........................................
      SUBROUTINE FLOWCN (
     1 GF    , GFI   , TFI   , SO    , GFSI  , GFSO , TO              ,
     2 GAUSS , WT    , CPHI  , GFM                                    ,
     3 X     , Y     , NOD   , NOP                                    ,
     4 BCV   , NCODV , NOPPV , MDFV  , NDNV                           ,
     5 NPE   , IR    , IF    , DT    , THETA , NDFV , NEM             ,
     6 NEQ    , NTRAN , NCARB   , NCYL                                   ,
     7 PRHS   , VPROP , IVIS                                            ,
     8 NSIZ   , NSTF  , NELM    , NNOD   , MAXFR                        ,
     9 R1     , ELF   , ELSTIF  , DMASS  , VHS                          ,
     1 LDEST  , NK    , EQ      , LHED   , LPIV                         ,
     2 JMOD   , QQ    , PVKOL   , NSB    , ISSB   , NSSB  , PBU         ,
     3 NCOD   , BC    , NOPP    , MDF    , NSDOF  , PPVL  , NBF         )
C...............................
      IMPLICIT REAL*8 (A-H,O-Z)
C...............................
      DIMENSION GF     ( NSIZ ) , GFI ( NSIZ ) , TFI ( NSIZ )
      DIMENSION SO     ( NSIZ ) , VHS ( NSIZ , 5)  , TO (NSIZ)
      DIMENSION GFSI   ( NSIZ ) , GFSO ( NSIZ )    , GFM ( NSIZ )
      DIMENSION CPHI   ( NSIZ )
      DIMENSION GAUSS  ( 4,4  ) , WT( 4,4 )
      DIMENSION VPROP  ( 30   )
      DIMENSION X      ( NNOD ) , Y ( NNOD )
      DIMENSION NOD    ( NELM   , 9 )             , NOP ( NELM , 9)
      DIMENSION BCV    ( NSIZ ) , NCODV ( NSIZ ) , NOPPV ( NSIZ )
      DIMENSION MDFV   ( NSIZ ) , NDNV   ( NSIZ )
      DIMENSION NSDOF  ( NSIZ , 4 ) , PPVL ( NSIZ )
      DIMENSION ISSB   ( NSIZ ) , NSSB   ( NSIZ , 3 ) , PBU (NSIZ)
C............................
      DIMENSION ELF    ( NSTF )   , ELSTIF ( NSTF,NSTF )
      DIMENSION ELF1   ( 18    )  , ELSTID ( 18    , 18 )
      DIMENSION DMASS  ( NSTF     , NSTF )
      DIMENSION LDEST  ( NSTF   )
      DIMENSION LHED   ( MAXFR )
      DIMENSION NK     ( NSTF   )
      DIMENSION LPIV   ( MAXFR )
      DIMENSION JMOD   ( MAXFR ) , QQ     ( MAXFR )
      DIMENSION PVKOL  ( MAXFR ) , R1 ( NSIZ )
      DIMENSION EQ     ( MAXFR    , MAXFR )
      DIMENSION BC     ( NSIZ   ) , NCOD ( NSIZ ) , NOPP ( NSIZ )
      DIMENSION MDF    ( NSIZ   )
C.............................
      CALL ARRZRF ( GF   , NEQ  )
      CALL ARRZRF ( R1   , NSIZ )
C...
C...
      CALL RSAVI  ( NOPP   , NOPPV   , NSIZ )
      CALL RSAVI  ( MDF    , MDFV    , NSIZ )
      CALL ARRZRI ( NCOD   , NSIZ )
      CALL ARRZRF ( BC     , NSIZ )
C       CALL RSAVE  ( BC   , BCV    , NSIZ )
C       CALL RSAVI  ( NCOD    , NCODV    , NSIZ )
C...
C
      DO 20 NE=1,NEM
         CALL  ARRZRF ( ELF  , NSTF )
         CALL  ARRZRF ( ELF1 , NSTF )
C.......................................................................
C                                                                       .
C     CHOOSE EITHER SUBROUTINE (STIFF) FOR CONTINUOUS PENALTY METHOD    .
C     OR STIFD FOR DISCRETE PENALTY METHOD                              .
```

```
C                                                                    .
C......................................................................
          CALL   STIFF ( NE    , NPE , GAUSS , WT   , ELSTIF ,NCYL    ,
     1                   VPROP , GFI , TFI   , VHS  , CPHI , NCARB     ,
     2                   NOD   , X   , Y     , IR , IF    , NELM  , NNOD,
     3                   NSIZ  , NSTF, IVIS  , GFSI,GFM                )

C CALCULATION OF ELEMENTAL LOAD VECTOR FOR BUBBLE  20,12,2004 M.PARVAZINIA

C         CALL    ELFVB ( NE    , NPE  , GAUSS , WT , ELF , GFI   , TFI,
C     1                   VPROP , IVIS , VHS   , AK , CP  , CPHI ,
C     2                   NOD   , X    , Y     , IR , IF  , NELM , NCYL,
C     3                   NNOD  , NSIZ , NSTF  , NSUPG  , GFSI, AKESI  )



C
C.. CALCULATION OF BOUNDARY INTEGRAL ( LINE INTEGRAL )
C
C   IF YOU WANT TO CALCULATE THE LINE INTEGRAL IN THE
C   SOLUTION SCHEME, ACTIVATE THE 'BUINTG' SUBROUTINE
C
C
C
          IF ( NSB.NE.0 ) THEN
             CALL BUINTG ( NE   , NOD  , X    , Y    , GAUSS , WT  , IF   ,
     1                     NELM , NNOD , NSIZ, ELF  , NSTF  , VHS , PRHS ,
     2                     VPROP, CPHI , TFI , IVIS, NCARB , NPE , NSB  ,
     3                     ISSB , NSSB , PBU , NCYL                      )
          ENDIF
C....
          IF ( NTRAN .EQ.2.OR.NTRAN.EQ.3 ) THEN
          IF ( THETA.EQ.1.0 ) THEN
          CALL  ARR2ZF ( ELSTID , NSTF )
          CALL  ARRZRF ( ELF1 , NSTF )
          ELSE
          CALL   STIFF ( NE    , NPE , GAUSS , WT   , ELSTID, NCYL      ,
     1                   VPROP , SO  , TO    , VHS  , CPHI , NCARB       ,
     2                   NOD   , X   , Y     , IR , IF    , NELM  , NNOD,
     3                   NSIZ  , NSTF, IVIS  , GFSO,GFM                )
C....................
          IF ( NSB.NE.0 ) THEN
          CALL BUINTG ( NE   , NOD  , X    , Y    , GAUSS , WT  , IF   ,
     1                  NELM , NNOD , NSIZ, ELF1, NSTF  , VHS , PRHS ,
     2                  VPROP, CPHI , TO   , IVIS, NCARB , NPE , NSB  ,
     3                  ISSB , NSSB , PBU , NCYL                      )
          ENDIF
C...
          ENDIF
             CALL MASS   ( NE , NPE , GAUSS , WT , DMASS, NOD  ,
     1                     X  , Y   , IR    , IF , NELM , NNOD , NSTF  ,
     2                     GFSI , VPROP , NSIZ  , NCYL                 )
             CALL ADDELF ( ELF , ELF1 , THETA , DT , NPE , NSTF , NDFV )
          CALL ELFC   ( NE     , NPE , DT     , THETA , ELSTID , ELF    ,
     1                  DMASS , SO   , NOD    , NELM  , NSIZ            )
          CALL ADDSF  ( NPE , DT , THETA , ELSTIF , DMASS , NDFV )

        ENDIF


C       CALL ARRZRF (ELF,NSTF)

C...
        CALL        PUTBCV ( NOD   , NCODV , BCV   , ELSTIF , ELF          ,
     1                       NSDOF, NE     , NSIZ  , NSTF   , NELM , NNOD ,
     2                       X     , Y     , PPVL  , NPE    , NBF  , VPROP,
     3                       GFI   , TFI   , IVIS  , VHS    , CPHI , NCARB,
     4                       GFSI  , IR    , IF    , GAUSS  , WT           )

        CALL FRONT
     1( ELSTIF    , ELF   , NE    , NOP   , NELM   , NSTF , LDEST , NK     ,
     2  MAXFR     , EQ    , LHED  , LPIV  , JMOD   , QQ   , PVKOL , GF     ,
     3  R1        , NCOD  , BC    , NOPP  , MDF    , NDNV , NSIZ  , NEM    ,
     4  NSIZ      , NEQ   , LCOL  , NELL  , NPE                            )
C...
 20     CONTINUE
C....................
        RETURN
C....................
        END
C...........................................................
C       CALCULATION OF ELEMENT STIFFNESS MATRIX  FLOW TERMS      .
C...........................................................
C
        SUBROUTINE STIFF ( NE     , NPE , GAUSS ,WT , ELSTIF  , NCYL     ,
     1                     VPROP , GFI , T      ,VHS  , CPHI   , NCARB    ,
     2                     NOD    , X   , Y      ,IR ,IF    , NELM   , NNOD,
     3                     NSIZ   , NSTF, IVIS   , GFSI   , GFM            )
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION  NOD   (NELM,9) , X (NNOD) , Y (NNOD)
        DIMENSION  VPROP ( 30   ) , VHS ( NSIZ , 5 ) , CPHI ( NSIZ )
        DIMENSION  GFI (NSIZ)    , T(NSIZ)   , GFSI ( NSIZ ) , GFM ( NSIZ )
        DIMENSION  AK11 (9,9) , AK12 (9,9) , AK21 (9,9) , AK22 (9,9)   ,
     1             S11  (9,9) , S12  (9,9) , S21  (9,9) , S22  (9,9)   ,
     2             DSIE (9)    , DSIK (9)    ,DSIKM(9),DSIEM(9),SIM(9)   ,
     3             XJ   (9)    , YJ   (9)    , AJ   (2,2) , AJI  (2,2)
        DIMENSION  ELSTIF (18,18) , GAUSS (7,7) , WT  (7,7) ,
     1             SI      (9)      , C11   (9,9) , C22 (9,9)

        DARCY=1E4


C.............
        CALL ARR2ZF ( AK11 , 9   )
        CALL ARR2ZF ( AK12 , 9   )
        CALL ARR2ZF ( AK21 , 9   )
        CALL ARR2ZF ( AK22 , 9   )
        CALL ARR2ZF ( S11  , 9   )
```

```
      CALL ARR2ZF ( S12   , 9    )
      CALL ARR2ZF ( S21   , 9    )
      CALL ARR2ZF ( S22   , 9    )
      CALL ARR2ZF ( C11   , 9    )
      CALL ARR2ZF ( C22   , 9    )
      CALL ARR2ZF ( ELSTIF , NSTF )
C.............
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
C     FULL INTEGRATION AND CONVECTION TERMS
C


      DO 24 KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO 24 KJ=1,IF
               ETA=GAUSS(KJ,IF)

               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1             DSIKM,DSIEM,SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)

C                CALL SHAPESH  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
C    1          DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD,
C   1                          AMU    , GAMAD , VPROP,  VHS, CPHI          ,
C   1                    G   , T ,NSIZ     , IVIS,NCARB)

               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
    1                        NPE , NE )
               CALL VISCOS ( AMU   , GAMAD , VPROP , NE    , VHS   , CPHI ,
    1                        NPE   , AKESI , ETA    , GFI   , T , NOD , X ,
    2                        Y     , NELM  , NNOD   , NSIZ , IVIS, NCARB )
               CALL UVN   ( UN    , VN , SI,SIM , NPE   , NE , GFI, NOD ,
    1                       NELM , NSIZ              )
               CALL UVN   ( UM    , VM , SI,SIM , NPE   , NE , GFM, NOD ,
    1                       NELM , NSIZ              )
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
               AMU = (FVAL*AMU +(1-FVAL)*VPROP(8))
               DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)


               IF ( NCYL.EQ.3) THEN
                  XR=0.0
                  DO KK=1,NPE
                     XR=XR+SI(KK)*XJ(KK)
                  ENDDO
                  XCC=XR
               ELSE
                  XCC=1.0
               ENDIF
               COEF= DET*WT(KI,IF)*WT(KJ,IF)*XCC
               DO 26 M=1,NPE
                  DSXM= DSIKM(M) * AJI(1,1) + DSIEM(M) * AJI(1,2)
                  DSYM= DSIKM(M) * AJI(2,1) + DSIEM(M) * AJI(2,2)

                  DO 26 N=1,NPE
                     DSXN= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                     DSYN= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)

                     AK11(M,N) = AK11(M,N) +
     1   (2*DSXN*DSXM+DSYN*DSYM+DARCY*SI(N)*SIM(M))*AMU*COEF

                     AK12(M,N) = AK12(M,N) +
     1                           AMU*DSXN*DSYM*COEF
                     AK21(M,N) = AK21(M,N) +
     1                           AMU*DSYN*DSXM*COEF
                    AK22(M,N) = AK22(M,N) +
     1   (2*DSYN*DSYM+DSXN*DSXM+DARCY*SI(N)*SIM(M))*AMU*COEF

C                     C11(M,N)= C11(M,N) + (UN-UM)*DEN * SI(M)*DSXN*COEF
C                     C22(M,N)= C22(M,N) + (VN-VM)*DEN * SI(M)*DSYN*COEF

 26             CONTINUE
 24           CONTINUE
C.....
C
C       REDUCED INTEGRATION
C
      CALL ARR2ZF ( S11   , 9    )
      CALL ARR2ZF ( S12   , 9    )
      CALL ARR2ZF ( S21   , 9    )
      CALL ARR2ZF ( S22   , 9    )
C..........
      DO 56 KI=1,IR
        AKESI=GAUSS(KI,IR)
        DO 56 KJ=1,IR
          ETA=GAUSS(KJ,IR)
            CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1     DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
            CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                   NPE , NE )
            CALL VISCOS ( AMU  , GAMAD , VPROP, NE   , VHS , CPHI  ,
     1                    NPE  , AKESI , ETA  , GFI  , T , NOD , X ,
     2                    Y    , NELM  , NNOD , NSIZ , IVIS, NCARB )
            CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
            AMU = (FVAL*AMU+(1-FVAL)*VPROP(8))
            IF ( NCYL.EQ.3) THEN
              XR=0.0
              DO KK=1,NPE
                XR=XR+SI(KK)*XJ(KK)
              ENDDO
              XCC=XR
            ELSE
              XCC=1.0
            ENDIF

            COEF= VPROP(15)*AMU*DET*WT(KI,IR)*WT(KJ,IR)*XCC
            DO 30 M=1,NPE
```

```
                  DSXM= DSIKM(M)  * AJI(1,1)  + DSIEM(M)  * AJI(1,2)
                  DSYM= DSIKM(M)  * AJI(2,1)  + DSIEM(M)  * AJI(2,2)

                  DO 30 N=1,NPE
                     DSXN= DSIK(N)  * AJI(1,1)  + DSIE(N)  * AJI(1,2)
                     DSYN= DSIK(N)  * AJI(2,1)  + DSIE(N)  * AJI(2,2)

                     IF ( NCYL.EQ.1 ) THEN
                        S11(M,N)= S11(M,N)  + DSXN*DSXM*COEF
                        S12(M,N)= S12(M,N)  + DSYN*DSXM*COEF
                        S21(M,N)= S21(M,N)  + DSXN*DSYM*COEF
                        S22(M,N)= S22(M,N)  + DSYN*DSYM*COEF
                     ELSE
                        S11(M,N)= S11(M,N)  + (DSXN+SI(N)/XR)*DSXM*COEF+
     1                               (SIM(M)/XR)*(DSXN+(SI(N)/XR))*COEF
                        S12(M,N)= S12(M,N)  + DSYN*DSXM*COEF+
     1                               (SIM(M)/XR)*DSYN*COEF
                        S21(M,N)= S21(M,N)  + (DSXN+SI(N)/XR)*DSYM*COEF
                        S22(M,N)= S22(M,N)  + DSYN*DSYM*COEF
                     ENDIF
 30               CONTINUE
 56      CONTINUE
C
C......................
            DO I=1,NPE
               DO J=1,NPE
                  AK11(I,J)=AK11(I,J)+ S11(I,J)+C11(I,J)+C22(I,J)
                  AK22(I,J)=AK22(I,J)+ S22(I,J)+C11(I,J)+C22(I,J)
                  AK12(I,J)=AK12(I,J)+ S12(I,J)
                  AK21(I,J)=AK21(I,J)+ S21(I,J)
               ENDDO
            ENDDO
C.......................
C
C     REORDERING  THE STIFFNESS MATRIX
C
      DO I=1,NPE
         M=2*I-1
         DO J=1,NPE
            N=2*J-1
            ELSTIF (M   , N   ) = AK11  (I,J)
            ELSTIF (M   , N+1 ) = AK12  (I,J)
            ELSTIF (M+1, N    ) = AK21  (I,J)
            ELSTIF (M+1, N+1 ) = AK22  (I,J)
        ENDDO
      ENDDO
C..................................................
      RETURN
      END


C..........................................
C     ELEMENT LOAD VECTOR FOR BUBBLE CALCULATION  20,12,2004 M.PARVAZINIA
.
C..........................................
```

```
      SUBROUTINE ELFVB ( NE    , NPE  , GAUSS , WT , ELF , GFI   , TFI,
     1                   VPROP , IVIS , VHS   , AK , CP  , CPHI ,
     2                   NOD   , X    , Y     , IR , IF  , NELM , NCYL,
     3                   NNOD  , NSIZ , NSTF  , NSUPG  , GFSI, AKESI  )


      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD( NELM,9) , X(NNOD)    , Y(NNOD)                  ,
     1          ELF(18)                   , GFI(NSIZ)   , TFI(NSIZ)    ,
     2          GAUSS(7,7)    , WT(7,7)                              ,
     3          DSIK(9) , DSIE(9) , SI(9), DSIKM(9),DSIEM(9),SIM(9) ,
     4          AJI(2,2)      , AJ(2,2)    , XJ(9)       , YJ(9)       ,
     5          VPROP (30)    , VHS ( NSIZ , 5 )       , GFSI ( NSIZ )
      DIMENSION CPHI ( NSIZ )
C .........................

C............B IS BUBBLE COEFFICIENT.............

      B=0.08

      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO

C.........................
      CALL ARRZRF ( ELF   , NSTF   )

C .........................
      DO 20 KI=1,IF
         AKESI=GAUSS(KI,IF)
         DO 20 KJ=1,IF
            ETA=GAUSS(KJ,IF)
            CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)

               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                                NPE , NE )

               CALL VISCOS ( AMU   , GAMAD , VPROP , NE    , VHS   , CPHI ,
     1                       NPE   , AKESI , ETA    , GFI   , T , NOD , X ,
     2                       Y     , NELM  , NNOD   , NSIZ , IVIS, NCARB )

            CALL UVN       ( UN   , VN , SI, SIM, NPE , NE   ,  GF,
     1                       NOD , NELM , NSIZ                    )
            CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )

            DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
            CPP = FVAL*CP   +(1-FVAL)*VPROP(11)
            AKK = FVAL*AK   +(1-FVAL)*VPROP(10)
            IF ( NCYL.EQ.3) THEN
               XR=0.0
               DO KK=1,NPE
                  XR=XR+SI(KK)*XJ(KK)
```

```
                ENDDO
                XCC=XR
             ELSE
                XCC=1.0
             ENDIF
             COEF= DET*WT(KI,IF)*WT(KJ,IF)*XCC


              IF (GAMAD.LT.0.1) THEN
                  GAMAD=1
               ELSE IF (GAMAD.GT.0.2) THEN
                  GAMAD=10*GAMAD
              END IF

                DO 25 M=1,NPE

 110           ELF(M)=ELF(M)+B*(1/GAMAD)*SIM(M)*(1-AKESI**2)*(1-ETA**2)*
     1                                                                COEF


 25          CONTINUE
 20     CONTINUE


        RETURN
        END


C.......................................
C       MASS MATRIX CALCULATION  (FLOW EQ.) .
C.......................................
        SUBROUTINE MASS ( NE , NPE , GAUSS , WT , DMASS, NOD       ,
     1                   X   , Y   , IR    , IF , NELM , NNOD , NSTF ,
     2                   GFSI , VPROP , NSIZ   , NCYL               )
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION GAUSS(7,7)  , WT(7,7) , DMASS  (18,18) , DM(9,9) ,
     1           SI(9)        , DSIE(9) , DSIK(9)        , XJ(9)   ,
     2           YJ(9)        , AJ(2,2) , AJI(2,2) ,
     3           DSIKM(9),DSIEM(9),SIM(9)
        DIMENSION NOD(NELM,9) , X(NNOD) , Y(NNOD)
        DIMENSION GFSI ( NSIZ ) , VPROP ( 30 )
        NDE= 2*NPE
        DO I= 1,NPE
           XJ(I)= X(NOD(NE,I))
           YJ(I)= Y(NOD(NE,I))
        ENDDO
C..........
        CALL ARR2ZF ( DMASS , NSTF )
        CALL ARR2ZF ( DM    , 9    )
C..........
        DO 24 KI=1,IF
              AKESI=GAUSS(KI,IF)
              DO 24 KJ=1,IF
                 ETA=GAUSS(KJ,IF)
                 CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1      DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
              CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                        NPE , NE )
              CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
C              DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
C             FOR DIMLESS DEN=1

              DEN=1
              IF ( NCYL.EQ.3) THEN
                 XR=0.0
                 DO KK=1,NPE
                    XR=XR+SI(KK)*XJ(KK)
                 ENDDO
                 XCC=XR
              ELSE
                 XCC=1.0
              ENDIF
              COEF= DET*WT(KI,IF)*WT(KJ,IF)*XCC
              DO 26 M=1,NPE
              DO 26 N=1,NPE
 26           DM(M,N)= DM(M,N)+DEN*SIM(M)*SI(N)*COEF
 24        CONTINUE
C.... REORDERING  THE MASS MATRIX                         .
      DO 44 I=1,NPE
         M=2*I-1
         DO 44 J=1,NPE
            N=2*J-1
            DMASS(M,N)=DM(I,J)
            DMASS(M,N+1)=0.0
            DMASS(M+1,N)=0.0
44          DMASS(M+1,N+1)=DM(I,J)
C.........................................
      CALL LUMP ( DMASS , NSTF , NDE )
C.........................................
      RETURN
      END
C...........................................................
C     ELEMENTAL LOAD VECTOR CALCULATION  ( FLOW EQ.)       .
C...........................................................
      SUBROUTINE ELFC ( NE    , NPE , DT    , THETA , ELSTIF , ELF   ,
     1                  DMASS , S   , NOD   , NELM  , NSIZ           )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD(NELM,9)
      DIMENSION S(NSIZ),ELSTIF(18,18),ELF(18),DMASS(18,18),DELTA(18)
      NDE=NPE*2
      DO I=1,NPE
         DELTA(2*I-1)=S(2*NOD(NE,I)-1)
         DELTA(2*I)=S(2*NOD(NE,I))
      ENDDO
      DO I=1,NDE
         DO J=1,NDE
         ELF(I)=ELF(I)+
     1          (DMASS(I,J)-DT*(1-THETA)*ELSTIF(I,J))*DELTA(J)
         ENDDO
      ENDDO
      RETURN
      END
C...............................................................
C     ADD STIFFNESS AND MASS MATRICES  SEE LHS OF (2.111) NASSEHI.
C...............................................................
      SUBROUTINE ADDSF ( NPE , DT , THETA , ELSTIF , DMASS , NDF )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ELSTIF(18,18),DMASS(18,18)
      NDE=NDF*NPE
      DO I=1,NDE
         DO J=1,NDE
            ELSTIF(I,J) = DMASS(I,J) + DT*THETA*ELSTIF(I,J)
         ENDDO
      ENDDO
      RETURN
      END
C................................................................
C     PUT THE FIRST AND THE THIRD DOF IN THE STIFFNESS EQUATIONS  .
C     ( PARTIAL SLIP CALCULATION )                                .
C................................................................
      SUBROUTINE PUTBCV ( NOD   , NCODZ , BCZ  , ELSTIF , ELF          ,
     1                    NSDOF, NE     , NSIZ , NSTF   , NELM , NNOD ,
     2                    X     , Y     , PPVL , NPE    , NBF  , VPROP,
     3                    GFI   , T     , IVIS , VHS    , CPHI , NCARB,
     4                    GFSI  , IR    , IF   , GAUSS  , WT          )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD ( NELM,9 ) , X( NNOD ) , Y( NNOD )
      DIMENSION VHS ( NSIZ , 5)  , CPHI ( NSIZ )
      DIMENSION BCZ (NSIZ)       , NCODZ (NSIZ)
      DIMENSION GFSI ( NSIZ )
      DIMENSION VPROP(30)        , GFI (NSIZ ) , T ( NSIZ )
      DIMENSION ELF ( NSTF ) , ELSTIF ( NSTF , NSTF )
      DIMENSION NSDOF ( NSIZ , 4 ) , PPVL ( NSIZ )
      DIMENSION DSIK(9) , DSIE(9) , SI(9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION XJ(9)   , YJ(9)
      DIMENSION AJ(2,2) , AJI(2,2)
      DIMENSION NP(3)
      DIMENSION GAUSS(7,7) , WT(7,7)
C.............
C....GVAL = 1.0D+300..........
C.............
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
C.... MODIFICATION OF STIFFNESS MATRIX AND LOAD VECTOR FOR 1ST DOF
C
      DO I= 1,NPE
         KBR=NOD(NE,I)
         DO J=1,2
            MBR= 2*KBR+J-2
            LBR= 2*I+J-2
            IF ( NCODZ(MBR).EQ. 1 ) THEN
               ELSTIF (LBR,LBR) =   1.0
```

```
              ELF    (LBR    ) = BCZ(MBR)


              DO K=1,NSTF
              IF  ( LBR.NE.K )  ELSTIF ( LBR,K ) = 0.0
              ENDDO
            ENDIF
          ENDDO
        ENDDO






C
C.... MODIFICATION OF STIFFNESS MATRIX AND LOAD VECTOR FOR 3RD DOF
C
      IF ( NBF.EQ.0 ) RETURN
      DO 100 IEN= 1,NBF
C
      IF ( NE.NE.NSDOF(IEN,4) ) GOTO 100
C
C.......................
      NP(3)= NOD ( NE , NSDOF ( IEN,3 ) )
      NP(2)= NOD ( NE , NSDOF ( IEN,2 ) )
      NP(1)= NOD ( NE , NSDOF ( IEN,1 ) )
C......
      DO 110 M = 1,NPE
           MG = NOD ( NE,M )
           IF ( MG.NE.NP(1).AND.MG.NE.NP(2).AND.MG.NE.NP(3) ) GOTO 110
C
C..  CALCULATION OF THE COMPONENTS OF THE UNIT VECTOR NORMAL TO THE
C    BOUNDARY
C
           CALL UTNML ( M , X , Y , NP , DNX , DNY, ELLGTH, NNOD ,
     1                  NSDOF , NSIZ   , IEN                       )
C.........................................
C    DETERMINING THE LOCAL COORDINATE VALUES
C    AND CALCULATION OF THE SHAPE FUNCTIONS AND THEIR DERIVATIVES
           CALL DAKE ( M,AKESI,ETA)
           CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1   DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
           CALL JACOB2 ( AJ  , AJI , DET , XJ , YJ , DSIK ,
     1                  DSIE, NPE , NE )
C
C..  CALCULATION OF THE SLIP PARAMETERS
C
           GII2 = VHS ( MG , 5 )
           GAMAD  = DSQRT ( 0.5*GII2 )
           GT = T(MG)
           CALL VISEQU ( AMU  , GAMAD  , GT , VPROP , IVIS )
           CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD,NELM,NSIZ)
C
C     MODIFICATION OF VISCOSITY TO INCLUDE THE EFFECT OF THE
C     EFFECTIVE FILLER VOULME FRACTION
```

```
C
            IF ( NCARB.EQ.2 ) THEN
               CPP =0.0
               DO I=1,NPE
                  CPP=CPP+ SI(I)*CPHI ( NOD(NE,I) )
               ENDDO
               CALL BOUN01 ( CPP )
               RELVIS = VPROP(21)+VPROP(22)*CPP
               AMU = AMU * RELVIS
            ENDIF
C
C
            AMU = FVAL*AMU+(1-FVAL)*VPROP(8)
            SLPARA = VPROP(6) * AMU
C           SLPARA = VPROP(6) * AMU / PPVL (MG)
C
C  INCORPORATION OF NAVIER'S SLIP CONDITION INTO THE WORKING EQUATIONS
C  X - DIRECTION
            MBR = 2*MG - 1
            LBR = 2*M  - 1
            IF ( NCODZ(MBR).NE.2) GOTO 114
C
C    MODIFICATION OF THE RIGHT HAND SIDE VECTOR
C
            ELF ( LBR ) = BCZ(MBR)
            DO 130 N = 1,NPE
                 KBRX = 2*N-1
                 KBRY = 2*N
                 DSX= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                 DSY= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)
                 ADDNNX = 2*DNY**2*DNX
                 ADDNNY = DNY*(DNY**2-DNX**2)
                 IGG = 0
                 IF ( LBR.EQ.KBRX ) IGG=1
                 ELSTIF(LBR,KBRX ) = SLPARA*(ADDNNX*DSX+ADDNNY*DSY)+IGG
                 ELSTIF(LBR,KBRY ) =-SLPARA*(ADDNNX*DSY-ADDNNY*DSX)
 130        CONTINUE
C
C
C
C  INCORPORATION OF NAVIER'S SLIP CONDITION INTO THE WORKING EQUATIONS
C  Y - DIRECTION
 114        MBR = 2*MG
            LBR = 2*M
            IF ( NCODZ(MBR).NE.2) GOTO 110
C
C    MODIFICATION OF THE RIGHT HAND SIDE VECTOR
C
            ELF ( LBR ) = BCZ(MBR)
            DO 135 N = 1,NPE
                  KBRX  = 2*N - 1
                  KBRY  = 2*N
                  DSX= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                  DSY= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)
                  ADDNNX = 2*DNX**2*DNY
                  ADDNNY = DNX*(DNY**2-DNX**2)
                  IGG = 0
                  IF ( LBR.EQ.KBRY ) IGG = 1
                  ELSTIF(LBR,KBRX )=-SLPARA*(ADDNNX*DSX+ADDNNY*DSY)
                  ELSTIF(LBR,KBRY )= SLPARA*(ADDNNX*DSY-ADDNNY*DSX)+IGG
 135         CONTINUE
C.....................................................
 110  CONTINUE
 100  CONTINUE
      RETURN
      END
C.....................................................
C    CALCULATION OF VISCOSITY                        .
C.....................................................
      SUBROUTINE VISCOS ( AMU    , GAMAD , VPROP, NE , VHS, CPHI      ,
     1                    NPE    , AKESI , ETA  , G  , T   , NOD  , X  ,
     2                    Y      , NELM  , NNOD , NSIZ     , IVIS,NCARB)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD   ( NELM,9) , X(NNOD) , Y(NNOD)
      DIMENSION G     ( NSIZ  ) , T(NSIZ) , CPHI ( NSIZ )
      DIMENSION VPROP ( 30    ) , VHS ( NSIZ , 5 )
      DIMENSION DSIK(9) , DSIE(9) , SI(9) ,DSIKM(9),DSIEM(9),SIM(9),
     1          XJ(9)   , YJ(9)             ,
     2          AJ(2,2) , AJI(2,2)
C....................................
      CALL SHAPE ( AKESI  ,ETA , DSIK , DSIE , SI , NPE ,
      1    DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
      CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1              NPE , NE )
      GUX= 0.0
      GUY= 0.0
      GVX= 0.0
      GVY= 0.0
      GT = 0.0
      GVR= 0.0
      DO I=1,NPE
         INN = NOD(NE,I)
         DFT = T   ( INN         )
         DFX = G   ( 2*INN - 1 )
         DFY = G   ( 2*INN       )
         DFR = VHS ( INN , 5 )
         RUX = DFX * ( DSIK(I) * AJI(1,1) + DSIE(I) * AJI(1,2) )
         RUY = DFX * ( DSIK(I) * AJI(2,1) + DSIE(I) * AJI(2,2) )
         RVX = DFY * ( DSIK(I) * AJI(1,1) + DSIE(I) * AJI(1,2) )
         RVY = DFY * ( DSIK(I) * AJI(2,1) + DSIE(I) * AJI(2,2) )
         GUX = GUX + RUX
         GUY = GUY + RUY
         GVX = GVX + RVX
         GVY = GVY + RVY
         GT  = GT  + DFT*SI(I)
         GVR = GVR + DFR*SI(I)
      ENDDO
C
C.... AII IS THE SECOND INVARIANT OF R.D.T BASED ON THE DIRECT CALCULATION
C     OF VELOCITY GRADIENT COMPONENTS
C
C.... GVR IS THE SECOND INVARIANT OF R.D.T CALCULATED BASED ON THE USE OF
C     VARIATIONAL RECOVERY METHOD
C
      AII=(2*GUX)**2+(2*GVY)**2+2*(GUY+GVX)**2
C       AII=2*(GUX+GVY)**2
      GAMAD=DSQRT(0.5*AII)
CC    PRINT*,GAMAD
      IF (GAMAD.LT.0.0001) GAMAD=0.0001

C...  GAMAD=DSQRT(0.5*GVR)
      CALL VISEQU ( AMU , GAMAD , GT , VPROP , IVIS )
C..........................................................
C     MODIFICATION OF VISCOSITY TO INCLUDE THE EFFECT OF THE .
C     EFFECTIVE FILLER VOLUME FRACTION                       .
C..........................................................
      IF ( NCARB.EQ.2 ) THEN
         CPP =0.0
         DO I=1,NPE
            CPP=CPP+ SI(I)*CPHI ( NOD(NE,I) )
         ENDDO
         CALL BOUN01 ( CPP )
         RELVIS = VPROP(21)+VPROP(22)*CPP
         AMU = AMU * RELVIS
      ENDIF
      RETURN
      END
C.................................................
C  VISCOSITY EQUATION SUBROUTINE                 .
C.................................................
      SUBROUTINE VISEQU ( AMU , GAMAD , GT , VPROP , IVIS )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION VPROP (30)
C...........................
      IF (GAMAD.EQ.0.0) THEN
          AMU=1
          RETURN
      ENDIF
             VPDMLS=423
      IF ( IVIS. EQ. 1 ) THEN
           AMU= GAMAD**(VPROP(2)-1)
     1          *DEXP ( -VPROP(3) * (GT- VPROP(4) ) )
      ELSEIF ( IVIS.EQ. 2) THEN
           AMU= ( 1+ (VPROP(5)*GAMAD)**2 )
     1           **( (VPROP(2)-1)/2 )
     2            *DEXP( -VPROP(3) * (GT-VPROP(4)) )
      ELSE
           WRITE ( 2 , 1000 )
           STOP
      ENDIF
 1000 FORMAT (1X,' ERROR IN VISCOSITY TYPE ')
```

```
      RETURN
      END
C......................................
C     SUBROUTINE DAKE
C......................................
      SUBROUTINE DAKE ( M,AKESI,ETA )
      IMPLICIT REAL*8 (A-H,O-Z)
      IF     ( M.EQ.1  ) THEN
         AKESI = -1
         ETA   = -1
      ELSEIF ( M.EQ.2  ) THEN
         AKESI =  1
         ETA   = -1
      ELSEIF ( M.EQ.3  ) THEN
         AKESI =  1
         ETA   =  1
      ELSEIF ( M.EQ.4  ) THEN
         AKESI = -1
         ETA   =  1
      ELSEIF ( M.EQ.5  ) THEN
         AKESI =  0
         ETA   = -1
      ELSEIF ( M.EQ.6  ) THEN
         AKESI =  1
         ETA   =  0
      ELSEIF ( M.EQ.7  ) THEN
         AKESI =  0
         ETA   =  1
      ELSEIF ( M.EQ.8  ) THEN
         AKESI = -1
         ETA   =  0
      ELSEIF ( M.EQ.9  ) THEN
         AKESI =  0
         ETA   =  0
      ENDIF
      RETURN
      END
C.......................................................
C    CALCULATION OF UN,VN                              .
C.......................................................
      SUBROUTINE UVN ( UN , VN   , SI ,SIM, NPE , NE , G , NOD ,
     1                 NELM , NSIZ                           )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD (NELM,9)
      DIMENSION  G   (NSIZ)
      DIMENSION  SI   (9), SIM (9)
      UN=0.0
      VN=0.0
      DO I=1,NPE
         UN = UN + SI(I) *G ( NOD(NE,I)*2-1 )
         VN = VN + SI(I) *G ( NOD(NE,I)*2   )

      ENDDO
      UN=1
      VN=1
      RETURN
      END


C.................................................................
C     CALCULATION OF ELEMENT STIFFNESS MATRIX FLOW TERMS         .
C     BASED ON THE DISCRETE PENALTY METHOD                       .
C.................................................................
C
      SUBROUTINE STIFD ( NE    , NPE , GAUSS ,WT  ,  ELSTIF  ,NCYL    ,
     1                   VPROP , GFI , T     ,VHS ,  CPHI , NCARB     ,
     2                   NOD   , X   , Y     ,IR ,IF    , NELM    , NNOD,
     3                   NSIZ  , NSTF, IVIS  , GFSI  ,GFM              )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD   (NELM,9) , X (NNOD) , Y (NNOD)
      DIMENSION  VPROP ( 30   ) , VHS ( NSIZ , 5 ) , CPHI ( NSIZ )
      DIMENSION  ELSTIF (18,18) , GAUSS (4,4) , WT  (4,4)
      DIMENSION  GFI (NSIZ)    , T(NSIZ) , GFSI ( NSIZ )
      DIMENSION  AK11 (9,9) , AK12 (9,9) , AK21 (9,9) , AK22 (9,9)
      DIMENSION  AKC  (9,9) , AKE  (9,9) , AKH  (9,9)
      DIMENSION  AKCL (9,9) , AKEL (9,9)
      DIMENSION   S11 (9,9) ,  S12 (9,9) ,  S21 (9,9) ,  S22 (9,9)
      DIMENSION  XJ   (9)   , YJ   (9)   , AJ   (2,2) , AJI  (2,2)
      DIMENSION  DSIK (9), DSIE (9) , SI (9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION  DSIKR(9)   , DSIER(9)   , SIR  (9)

      DARCY=1E5

C.............
      CALL ARR2ZF ( AK11 , 9   )
      CALL ARR2ZF ( AK12 , 9   )
      CALL ARR2ZF ( AK21 , 9   )
      CALL ARR2ZF ( AK22 , 9   )
      CALL ARR2ZF ( S11  , 9   )
      CALL ARR2ZF ( S12  , 9   )
      CALL ARR2ZF ( S21  , 9   )
      CALL ARR2ZF ( S22  , 9   )
      CALL ARR2ZF ( AKC  , 9   )
      CALL ARR2ZF ( AKE  , 9   )
      CALL ARR2ZF ( AKH  , 9   )
      CALL ARR2ZF ( AKCL , 9   )
      CALL ARR2ZF ( AKEL , 9   )
      CALL ARR2ZF ( ELSTIF , NSTF )
C.............
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
      NPR=IR*IR
C
C
C     INTEGRATION
C
```

```
      DO KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1         DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL SHAPE  ( AKESI , ETA , DSIKR, DSIER, SIR, NPR ,
      1         DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                      NPE , NE )
               CALL VISCOS ( AMU   , GAMAD , VPROP , NE   , VHS , CPHI   ,
     1                       NPE   , AKESI , ETA   , GFI  , T , NOD , X ,
     2                       Y     , NELM  , NNOD  , NSIZ , IVIS ,NCARB )
               CALL UVN    ( UN    , VN , SI ,SIM, NPE   , NE , GFI, NOD ,
     1                     NELM , NSIZ             )
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
               AMU = FVAL*AMU+(1-FVAL)*VPROP(8)
               DETCOF= DET*WT(KI,IF)*WT(KJ,IF)
               GVL   = VPROP(15)*AMU
               DO  M=1,NPE
                   DSXM= DSIKM(M)  * AJI(1,1)  + DSIEM(M)  * AJI(1,2)
                   DSYM= DSIKM(M)  * AJI(2,1)  + DSIEM(M)  * AJI(2,2)
                   DO  N=1,NPE
                       DSXN= DSIK(N)  * AJI(1,1)  + DSIE(N)  * AJI(1,2)
                       DSYN= DSIK(N)  * AJI(2,1)  + DSIE(N)  * AJI(2,2)
                       AK11  (M,N)  = AK11  (M,N ) +
     1                 ((2*DSXM*DSXN+DSYM*DSYN+DARCY*SI(N)*SIM(M))*AMU +
     2                                ( VPROP(14)*UN*SIM(M)*DSXN)    +
     3                              ( VPROP(14)*VN*SIM(M)*DSYN)  )*DETCOF
                       AK22  (M,N)  = AK22  (M,N ) +
     1                 ((2*DSYM*DSYN+DSXM*DSXN+DARCY*SI(N)*SIM(M))*AMU +
     2                              ( VPROP(14)*UN*SIM(M)*DSXN)          +
     3                              ( VPROP(14)*VN*SIM(M)*DSYN)  )*DETCOF
                       AK12  (M,N)  = AK12  (M,N )+DSYM*DSXN*AMU*DETCOF
                       AK21  (M,N)  = AK21  (M,N )+DSYN*DSXM*AMU*DETCOF
                   ENDDO
               ENDDO
C.......................................................................
               DO  M=1,NPE
                   DSXM= DSIKM(M)  * AJI(1,1)  + DSIEM(M)  * AJI(1,2)
                   DSYM= DSIKM(M)  * AJI(2,1)  + DSIEM(M)  * AJI(2,2)
                   DO  N=1,NPR
                       AKC  (M,N)  = AKC  (M,N)+(-1)*DSXM*SIR(N)*DETCOF
                       AKE  (M,N)  = AKE  (M,N)+(-1)*DSYM*SIR(N)*DETCOF
                       AKCL(M,N)  = AKCL(M,N)+(-1)*DSXM*SIR(N)*DETCOF*GVL
                       AKEL(M,N)  = AKEL(M,N)+(-1)*DSYM*SIR(N)*DETCOF*GVL
                   ENDDO
               ENDDO
C.......................................................................
               DO  M=1,NPR
                   DO  N=1,NPR
                       AKH  (M,N)  = AKH  (M,N)+SIR(M)*SIR(N)*DETCOF
                   ENDDO
               ENDDO
C.......................................................................
```

```
               ENDDO
        ENDDO
C
C       DIAGONALIZATION AND INVERSION OF MATRIX (H)
C
        DO I=1,NPR
           SUM=0.0
           DO J=1,NPR
              SUM = SUM + AKH (I,J)
              AKH (I,J) =0.0
           ENDDO
           IF ( SUM.NE.0.0 ) AKH (I,I) = 1.0/SUM
        ENDDO
C
C       TRANSPOSE OF MATRICES AKCL AND AKEL
C
        CALL TRAP ( AKCL , NPE )
        CALL TRAP ( AKEL , NPE )
C
C
C
        CALL MULT3 ( S11 , AKC , AKH , AKCL , NPE  )
        CALL MULT3 ( S12 , AKC , AKH , AKEL , NPE  )
        CALL MULT3 ( S21 , AKE , AKH , AKCL , NPE  )
        CALL MULT3 ( S22 , AKE , AKH , AKEL , NPE  )
C
C
C
        DO I=1,NPE
           DO J=1,NPE
              AK11 (I,J) = AK11(I,J) + S11 (I,J)
              AK12 (I,J) = AK12(I,J) + S12 (I,J)
              AK21 (I,J) = AK21(I,J) + S21 (I,J)
              AK22 (I,J) = AK22(I,J) + S22 (I,J)
           ENDDO
        ENDDO
C
C       REORDERING  THE STIFFNESS MATRIX
C
        DO I=1,NPE
           M=2*I-1
           DO J=1,NPE
              N=2*J-1
              ELSTIF (M   , N   ) = AK11 (I,J)
              ELSTIF (M   , N+1 ) = AK12 (I,J)
              ELSTIF (M+1, N    ) = AK21 (I,J)
              ELSTIF (M+1, N+1 ) = AK22 (I,J)
          ENDDO
        ENDDO
C..........................................................
        RETURN
        END
C................................................
C   MUTIPLICATION OF THREE MATRICES              .
C................................................
```

```
      SUBROUTINE MULT3 ( R , A , B , C , NPE  )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  G(9,9),R(9,9),A(9,9),B(9,9),C(9,9)
      DO I=1,NPE
         DO J=1,NPE
            R(I,J)=0.0
            G(I,J)=0.0
         ENDDO
      ENDDO
      CALL MULT2 ( G , A , B , NPE )
      CALL MULT2 ( R , G , C , NPE )
      RETURN
      END
C..............................................
C   MUTIPLICATION OF TWO   MATRICES           .
C..............................................
      SUBROUTINE MULT2 ( C , A , B , NPE  )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  A(9,9),B(9,9),C(9,9)
      DO I=1,NPE
         DO J=1,NPE
            C(I,J)=0.0
         ENDDO
      ENDDO
      DO I=1,NPE
         DO  J=1,NPE
             DO K=1,NPE
                C(I,J)=C(I,J)+A(I,K)*B(K,J)
             ENDDO
         ENDDO
      ENDDO
      RETURN
      END
C....................................................
C    CALCULATION OF PRESSURE                        .
C    IN  DISCRETE PENALTY METHOD                    .
C....................................................
      SUBROUTINE PRESD ( NE     , NPE , GAUSS ,WT , CPHI                 ,
     1                   VPROP , GF   , T      , X , Y     , NCARB       ,
     2                   NOD   , IR   , IF     , NELM , NNOD  , NEM      ,
     3                   NSIZ  , NSTF, IVIS , VHS   , PSOUT , GFS        )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD (NELM,9)    , X(NNOD)  , Y(NNOD)
      DIMENSION  VPROP ( 30    ) , VHS ( NSIZ , 5) , CPHI ( NSIZ )
      DIMENSION  GAUSS (7,7) , WT  (7,7)
      DIMENSION  GF (NSIZ)    , T(NSIZ)     , GFS ( NSIZ )
      DIMENSION  AKCL (9,9) , AKEL (9,9) , AKH  (9,9)
      DIMENSION   S11 (9,9) ,  S12 (9,9)
      DIMENSION  AKUU (9)    ,  AKVV(9)
      DIMENSION  AKP1 (9)    ,  AKP2(9)
      DIMENSION  XJ    (9)    , YJ    (9)    , AJ   (2,2) , AJI  (2,2)
      DIMENSION  DSIK (9) , DSIE (9), SI (9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION  DSIKR(9)    , DSIER(9)    , SIR  (9)
      DIMENSION  PSOUT ( NSIZ )
C.............
```

```
      DO 1000 NE=1,NEM
C.............
      CALL ARR2ZF ( S11  , 9   )
      CALL ARR2ZF ( S12  , 9   )
      CALL ARR2ZF ( AKH  , 9   )
      CALL ARR2ZF ( AKCL , 9   )
      CALL ARR2ZF ( AKEL , 9   )
      CALL ARRZRF ( AKUU , 9   )
      CALL ARRZRF ( AKVV , 9   )
      CALL ARRZRF ( AKP1 , 9   )
      CALL ARRZRF ( AKP2 , 9   )
C....................
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
      NPR=IR*IR
C
C
C     INTEGRATION
C
      DO KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1        DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
               CALL SHAPE  ( AKESI , ETA , DSIKR, DSIER, SIR, NPR ,
     1        DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                       NPE , NE )
               CALL VISCOS ( AMU  , GAMAD , VPROP , NE   , VHS , CPHI  ,
     1                        NPE  , AKESI , ETA   , GF   , T , NOD , X ,
     2                        Y    , NELM  , NNOD  , NSIZ , IVIS ,NCARB )
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFS , NOD, NELM, NSIZ )
C              AMU = FVAL*AMU+(1-FVAL)*VPROP(8)
               DETCOF= DET*WT(KI,IF)*WT(KJ,IF)
               GVL   = VPROP(15)*AMU
C.........................................................
               DO  M=1,NPE
                   DSXM= DSIK(M) * AJI(1,1) + DSIE(M) * AJI(1,2)
                   DSYM= DSIK(M) * AJI(2,1) + DSIE(M) * AJI(2,2)
                   DO  N=1,NPR
                       AKCL(M,N) = AKCL(M,N)+(-1)*DSXM*SIR(N)*DETCOF*GVL
                       AKEL(M,N) = AKEL(M,N)+(-1)*DSYM*SIR(N)*DETCOF*GVL
                   ENDDO
               ENDDO
C.........................................................
               DO  M=1,NPR
                   DO  N=1,NPR
                       AKH  (M,N) = AKH  (M,N)+SIR(M)*SIR(N)*DETCOF
                   ENDDO
               ENDDO
C.........................................................
```

```
            ENDDO
      ENDDO
C
C     DIAGONALIZATION AND INVERSION OF MATRIX (H)
C
      DO I=1,NPR
         SUM=0.0
         DO J=1,NPR
            SUM = SUM + AKH (I,J)
            AKH (I,J) =0.0
         ENDDO
         IF ( SUM.NE.0.0 ) AKH (I,I) = 1.0/SUM
      ENDDO
C
C     TRANSPOSE OF MATRICES AKCL AND AKEL
C
      CALL TRAP ( AKCL , NPE )
      CALL TRAP ( AKEL , NPE )
C
C
C
      CALL MULT2 ( S11 , AKH , AKCL , NPE )
      CALL MULT2 ( S12 , AKH , AKEL , NPE )
C
C
C
      DO I=1,NPE
         NDFFX =2*NOD(NE,I)-1
         NDFFY =2*NOD(NE,I)
         AKUU(I) = GF ( NDFFX )
         AKVV(I) = GF ( NDFFY )
      ENDDO
      DO I=1,NPE
         DO J=1,NPE
            AKP1 (I)= AKP1(I) + S11(I,J)*AKUU(J)
            AKP2 (I)= AKP2(I) + S12(I,J)*AKVV(J)
         ENDDO
      ENDDO
      DO I=1,NPE
         PSOUT ( NOD(NE,I))=PSOUT ( NOD(NE,I))+AKP1(I)+AKP2(I)
      ENDDO
      AKESI = 0.0
      ETA   = 0.0
      CALL SHAPE  ( AKESI , ETA , DSIKR, DSIER, SIR, NPR ,
      1     DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
      DO I=1,NPR
          PSOUT ( NOD(NE,9))=PSOUT ( NOD(NE,9))+SIR(I)*(AKP1(I)+AKP2(I))
      ENDDO
1000  CONTINUE
C.................................................
      RETURN
      END
C..............................................
C                                             .
C     CALCULATION OF BOUNDARY INTEGRALS IN    .
```

```
C     9-NODED ELEMENTS                                         .
C                                                              .
C..............................................................
      SUBROUTINE BUINTG ( NE   , NOD  , X    , Y    , GAUSS , WT   , IF   ,
     1                    NELM , NNOD , NSIZ, ELF  , NSTF  , VHS  , PRHS ,
     2                    VPROP, CPHI , T    , IVIS, NCARB , NPE  , NSB  ,
     3                    ISSB , NSSB , PBU  , NCYL                       )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD ( NELM,9 ) , X (NNOD) , Y(NNOD)
      DIMENSION  GAUSS (7,7)    , WT(7,7)
      DIMENSION  SI(3)          , DSI (3)
      DIMENSION  NP ( 3,4)      , ELF ( NSTF )
      DIMENSION  NQ ( 4,3)
      DIMENSION  VHS ( NSIZ , 5 ) , PRHS ( NSIZ )
      DIMENSION  VPROP ( 30 ) , CPHI ( NSIZ ) , T ( NSIZ )
      DIMENSION  ISSB  ( NSIZ ) , NSSB  ( NSIZ , 3 ) , PBU (NSIZ)
C.......................................................................
      NQ(1,3)=2
      NQ(1,2)=5
      NQ(1,1)=1
      NQ(2,3)=3
      NQ(2,2)=6
      NQ(2,1)=2
      NQ(3,3)=4
      NQ(3,2)=7
      NQ(3,1)=3
      NQ(4,3)=1
      NQ(4,2)=8
      NQ(4,1)=4
      DO I=1,4
         DO J=1,3
            NP (4-J,I) = NOD ( NE , NQ (I,4-J) )
         ENDDO
      ENDDO
      CALL ARRZRF  ( ELF    , NSTF )
      DO L=1,4
C
C... SUBSTITUTION OF PRESSURE WITH BOUNDARY VALUES
C
         IPCAL =0
         IF (NSB.NE.0) THEN
            DO K=1,NSB
               IF (ISSB(K).EQ.NE) THEN
                     IF (NSSB ( K , 1 ).EQ.NQ (L,1).AND.
     1                   NSSB ( K , 2 ).EQ.NQ (L,2).AND.
     2                   NSSB ( K , 3 ).EQ.NQ (L,3)     ) THEN
                           PRHS ( NP ( 1 , L )) = PBU (K)
                           PRHS ( NP ( 2 , L )) = PBU (K)
                           PRHS ( NP ( 3 , L )) = PBU (K)
                           IPCAL = 1
                     ENDIF
               ENDIF
            ENDDO
         ENDIF
C
```

```
C.................................................................
C
         DO K = 1,IF
            AKESI = GAUSS (K,IF)
            CALL LAGSH1 ( AKESI , SI , DSI )
            DSIX =0.0
            DSIY =0.0
            DO I= 1,IF
               DSIX = DSIX + DSI(I)*X( NP(I,L) )
               DSIY = DSIY + DSI(I)*Y( NP(I,L) )
            ENDDO
            IF ( NCYL.EQ.3) THEN
               XR=0.0
               DO KK=1,IF
                  XR=XR+SI(KK)*X(NP(KK,L))
               ENDDO
               XCC=XR
            ELSE
               XCC=1.0
            ENDIF
            DUX  = 0.0
            DUY  = 0.0
            DVX  = 0.0
            DVY  = 0.0
            PRS  = 0.0
            GII2 = 0.0
            GT   = 0.0
            CPP  = 0.0
C
C.. CALCULATION OF VELOCITY GRADIENTS, SHEAR RATE, TEMPERATURE AND
C.. EFVF AT INTEGRATION POINTS
C
            DO I=1,IF
               DUX  = DUX + SI(I)*VHS  ( NP(I,L) , 1 )
               DUY  = DUY + SI(I)*VHS  ( NP(I,L) , 2 )
               DVX  = DVX + SI(I)*VHS  ( NP(I,L) , 3 )
               DVY  = DVY + SI(I)*VHS  ( NP(I,L) , 4 )
               GII2 = GII2+ SI(I)*VHS  ( NP(I,L) , 5 )
               PRS  = PRS + SI(I)*PRHS ( NP(I,L) )
               GT   = GT  + SI(I)*T    ( NP(I,L) )
               CPP  = CPP + SI(I)*CPHI ( NP(I,L) )
            ENDDO
            IF ( GII2.LT.0.0 ) GII2=0.0
            GAMAD  = DSQRT ( 0.5*GII2 )
C
C.. CALCULATION OF VISCOSITY
C
            CALL VISEQU ( AMU  , GAMAD  , GT , VPROP , IVIS )
C
C.. MODIFICATION OF VISCOSITY FOR EFFECT OF THE EFVF
C
            IF ( NCARB.EQ.2 ) THEN
                CALL BOUN01 ( CPP )
                RELVIS = VPROP(21)+VPROP(22)*CPP
                AMU = AMU * RELVIS
            ENDIF
C
C     CALCULATION OF ELEMENT LENGTH AND THE COMPONENTS OF THE
C     UNIT VECTOR NORMAL TO THE BOUNDARY
C
            ELLGTH = DSQRT ( DSIX **2 + DSIY  **2 )
            DCELL  =  DSIX  / ELLGTH
            DCELM  =  DSIY  / ELLGTH
            DNX    =  DCELM
            DNY    = -DCELL
            ELLGTH = 2.0 * ELLGTH
            DJACOB = ELLGTH / 2.0
            DO M=1,IF
              NDFX = 2 * NQ (L,M) - 1
              NDFY = 2 * NQ (L,M)
              IF (IPCAL.EQ.0) THEN
                ELXX= 0.0
                ELYY= 0.0
              ELSE
                ELXX= (2*AMU*DUX*DNX+AMU*(DUY+DVX)*DNY-PRS*DNX)*
     1                   SI(M)*DJACOB*WT(K,IF)*XCC
                ELYY= (2*AMU*DVY*DNY+AMU*(DUY+DVX)*DNX-PRS*DNY)*
     1                  SI(M)*DJACOB*WT(K,IF)*XCC
              ENDIF
              ELF ( NDFX ) = ELF ( NDFX ) +  ELXX
              ELF ( NDFY ) = ELF ( NDFY ) +  ELYY
            ENDDO
          ENDDO
        ENDDO
        RETURN
        END
C.........................................
C    SOLUTION OF ENERGY EQUATION             .
C.........................................
        SUBROUTINE TMPRUR (
     1  TF    , TFI   , GF      , TO    , SO                            ,
     2  GAUSS , WT    , VHS     , GFSI  , GFSO   , GFM                  ,
     3  X     , Y     , NOD     , NOP   , CPHI                          ,
     4  BCT   , NCODT , NOPPT   , MDFT  , NDNT                          ,
     5  NPE   , IR    , IF      , DT    , THETA  , NDFT , NEM           ,
     6  NET   , NNM   , NTRAN   , NSUPG , NCARB                         ,
     7  AK    , CP    , VPROP                                           ,
     8  NSIZ  , NSTF  , NELM    , NNOD  , MAXFR  , IVIS                 ,
     9  R1    , ELF   , ELSTIF  , DMASS , ELF1                          ,
     1  LDEST , NK    , EQ      , LHED  , LPIV                          ,
     2  JMOD  , QQ    , PVKOL   , NCYL                                  ,
     3  NCOD  , BC    , NOPP    , MDF                                   )
C.............................
        IMPLICIT REAL*8 (A-H,O-Z)
C.............................
        DIMENSION TF     ( NSIZ ) , TFI ( NSIZ ), GF   ( NSIZ )
        DIMENSION TO     ( NSIZ ) , SO   ( NSIZ ), CPHI ( NSIZ )
        DIMENSION GFSI   ( NSIZ ) , GFSO ( NSIZ ) , GFM ( NSIZ )
        DIMENSION GAUSS ( 7,7   ) , WT( 7,7 )
        DIMENSION VPROP ( 30    ) , VHS ( NSIZ , 5 )
```

```
      DIMENSION X      ( NNOD ) , Y ( NNOD )
      DIMENSION NOD    ( NELM   , 9 )              , NOP ( NELM , 9)
      DIMENSION BCT    ( NSIZ ) , NCODT ( NSIZ ) , NOPPT ( NSIZ )
      DIMENSION MDFT   ( NSIZ ) , NDNT  ( NSIZ )
C.............................
      DIMENSION ELF    ( NSTF )  , ELSTIF ( NSTF,NSTF )
      DIMENSION DMASS  ( NSTF    , NSTF ) , ELF1 ( NSTF )
      DIMENSION LDEST  ( NSTF   )
      DIMENSION LHED   ( MAXFR )
      DIMENSION NK     ( NSTF   )
      DIMENSION LPIV   ( MAXFR )
      DIMENSION JMOD   ( MAXFR ) , QQ    ( MAXFR )
      DIMENSION PVKOL  ( MAXFR ) , R1 ( NSIZ )
      DIMENSION EQ     ( MAXFR   , MAXFR )
      DIMENSION BC     ( NSIZ   ) , NCOD ( NSIZ ) , NOPP ( NSIZ )
      DIMENSION MDF    ( NSIZ   )
C..........................
      CALL ARRZRF ( TF , NET   )
      CALL ARRZRF ( R1 , NSIZ )
C ....
C ....
      CALL RSAVE ( BC   , BCT   , NSIZ )
      CALL RSAVI ( NCOD , NCODT , NSIZ )
      CALL RSAVI ( NOPP , NOPPT , NSIZ )
      CALL RSAVI ( MDF  , MDFT  , NSIZ )
C.....
      DO 34 NE=1,NEM
         CALL  ARRZRF ( ELF  , NSTF )
         CALL  ARRZRF ( ELF1 , NSTF )
         CALL STIFFT  ( NE    , NPE   , GAUSS , WT  , AK  , ELSTIF , GF,
     1                  CP    , NSUPG , NOD   , X   , Y   , IR          ,
     2                  IF    , NELM  , NNOD  , NSIZ  , NSTF , GFSI     ,
     3                  VPROP , GFM   , NCYL                            )
         CALL ELFTS   ( NE    , NPE   , GAUSS , WT  , ELF , GF   , TFI ,
     1                  VPROP , IVIS  , VHS   , AK  , CP  , CPHI ,
     2                  NOD   , X     , Y     , IR  , IF  , NELM , NCYL,
     3                  NNOD  , NSIZ  , NSTF  , NSUPG  , GFSI , NCARB  )
         IF (NTRAN.NE.1) THEN
            CALL MASSW  ( NE    , NPE   , GAUSS , WT , CP   ,
     1                    DMASS , NOD   , X     , Y  , IR   , IF ,
     2                    NELM  , NNOD  , NSTF  , NSUPG , AK, GF ,
     3                    NSIZ  , GFSI  , VPROP , NCYL               )
            CALL ELFTS ( NE    , NPE , GAUSS , WT , ELF1 , SO    , TO  ,
     1                   VPROP , IVIS, VHS   , AK , CP   , CPHI ,
     2                   NOD   , X   , Y     , IR , IF   , NELM , NCYL,
     3                   NNOD  , NSIZ  , NSTF, NSUPG  , GFSO , NCARB  )
            CALL ADDELF ( ELF , ELF1 , THETA , DT , NPE , NSTF , NDFT )
            CALL ELFT   ( NE    , NPE  , DT   , THETA , ELSTIF , ELF   ,
     1                    DMASS , TO   , NOD , NELM ,
     2                    NSIZ                                  )
            CALL ADDSF  ( NPE,DT,THETA,ELSTIF,DMASS,NDFT)
         ENDIF
         CALL FRONT
     1( ELSTIF    , ELF    , NE   , NOP  , NELM   , NSTF , LDEST , NK     ,
     2  MAXFR     , EQ     , LHED , LPIV , JMOD   , QQ   , PVKOL , TF     ,
     3  R1        , NCOD   , BC    , NOPP , MDF    , NDNT , NSIZ   , NEM    ,
     4  NSIZ      , NET    , LCOL  , NELL , NPE                             )
  34  CONTINUE
C...................
      RETURN
C...................
      END
C.......................................
C    STIFFNES MATRIX FOR TEMPEREATURE         .
C.......................................
      SUBROUTINE  STIFFT ( NE    , NPE    , GAUSS, WT    , AK , ELSTIF, GF,
     1                     CP    , NSUPG , NOD   , X     , Y  , IR         ,
     2                     IF    , NELM  , NNOD  , NSIZ  , NSTF            ,
     3                     GFSI  , VPROP , GFM   , NCYL                     )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD ( NELM,9 ) , X (NNOD) , Y(NNOD)                   ,
     1          GAUSS (7,7), WT(7,7)    , ELSTIF(18,18) , GF(NSIZ)   ,
     2          AKC (9,9)   , AKV (9,9) , AKW (9,9)      , AKWD(9,9) ,
     3          DSIK(9) , DSIE(9) , SI(9) ,DSIKM(9),DSIEM(9),SIM(9),
     4          DSKK(9)      , DSEE(9)    , DSKE(9)    ,  GFM (NSIZ)   ,
     5          XJ(9)         , YJ(9)                                   ,
     6          AJI(2,2)     , AJ(2,2) ,    DSIK1(9) , DSIE1(9)         ,
     7          GFSI ( NSIZ ) , VPROP (15)
C.............................................................
      CALL ARR2ZF ( AKC   , 9 )
      CALL ARR2ZF ( AKV   , 9 )
      CALL ARR2ZF ( AKW   , 9 )
      CALL ARR2ZF ( AKWD , 9 )
      CALL ARR2ZF ( ELSTIF , NSTF )
C.........
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
C ...  CALCULATION OF CONDUTION AND STANDARD CONVECTION TERMS
C


      DO 14 KI=1,IF
         AKESI=GAUSS(KI,IF)
         DO 14 KJ=1,IF
            ETA=GAUSS(KJ,IF)
            CALL SHAPEB   ( AKESI , ETA , DSIK1 , DSIE1 , SI , NPE ,
     1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD ,
     2  DSIK, DSIE,UN,VN)
C             CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
C    1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
            CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                       NPE , NE )
            CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
            CALL UVN   ( UN    , VN , SI ,SIM, NPE   , NE , GF , NOD ,
     1                  NELM , NSIZ              )
            CALL UVN   ( UM    , VM , SI,SIM , NPE   , NE , GFM, NOD ,
     1                  NELM , NSIZ              )
```

```
            AKK = FVAL*AK   +(1-FVAL)*VPROP(10)
            DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
            CPP = FVAL*CP   +(1-FVAL)*VPROP(11)

      DIFF=AKK
      CONV=DEN*CPP

C      PRINT*,DIFF,CONV


            IF ( NCYL.EQ.3) THEN
               XR=0.0
               DO KK=1,NPE
                  XR=XR+SI(KK)*XJ(KK)
               ENDDO
               XCC=XR
            ELSE
               XCC=1.0
            ENDIF
            COEF= DET*WT(KI,IF)*WT(KJ,IF)*XCC
            DO 16 M=1,NPE
               DSXM= DSIKM(M) * AJI(1,1) + DSIEM(M) * AJI(1,2)
               DSYM= DSIKM(M) * AJI(2,1) + DSIEM(M) * AJI(2,2)
               DO 16 N=1,NPE

                        DSXN=  DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                  DSYN=  DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)

                  DSXN1= DSIK1(N) * AJI(1,1) + DSIE1(N) * AJI(1,2)
                  DSYN1= DSIK1(N) * AJI(2,1) + DSIE1(N) * AJI(2,2)

        AKC(M,N)= AKC(M,N)+AKK*(DSXM*DSXN+DSYM*DSYN)*COEF
                  AKV(M,N)= AKV(M,N)+DEN*CPP*
CCC     1                ((UN-UM)*SIM(M)*DSXN1+(VN-VM)*SIM(M)*DSYN1)*COEF
     1      (SIM(M)*DSXN1+SIM(M)*DSYN1)*COEF+0*SI(N)*SIM(M)*COEF

c.........for time  here y is time direction .........................

c        AKC(M,N)= AKC(M,N)+DIFF*(DSXM*DSXN)*COEF
c                  AKV(M,N)= AKV(M,N)+
c     1   (SIM(M)*DSYN1+CONV*SIM(M)*DSXN1)*COEF+0*SI(N)*SIM(M)*COEF

 16          CONTINUE
 14     CONTINUE
C..
C..
      IF ( NSUPG .EQ. 1 ) GOTO 100
C
C ... CALCULATION OF UPWIND TERMS
C
C... CALCULATION OF SU/PG CONVECTION TERMS
C
      DO 24 KI=1,IF
         AKESI=GAUSS(KI,IF)
         DO 24 KJ=1,IF
```

```
          ETA=GAUSS(KJ,IF)
          CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
          IF (NPE.EQ.9) THEN
          CALL SH9DD   ( AKESI , ETA , DSKK , DSEE,  DSKE      )
          ELSE
          CALL SH4DD   ( AKESI , ETA , DSKK , DSEE,  DSKE      )
          END IF
          CALL JACOB2  ( AJ   , AJI , DET , XJ , YJ , DSIK , DSIE ,
    1                    NPE , NE )
          CALL UVN     ( UN   , VN , SI,SIM , NPE   , NE     ,  GF,
    1                    NOD , NELM , NSIZ                  )
          CALL UVN   ( UM    , VM , SI ,SIM, NPE   , NE , GFM, NOD ,
    1                  NELM , NSIZ              )
          CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
          AKK = FVAL*AK   +(1-FVAL)*VPROP(10)
          DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
          CPP = FVAL*CP   +(1-FVAL)*VPROP(11)
C..................................................................
          AA1 = AJ ( 1,1 )
          AA2 = AJ ( 1,2 )
          AA3 = 0.0
          AA4 = 0.0
          AA5 = 0.0
          BB1 = AJ ( 2,1 )
          BB2 = AJ ( 2,2 )
          BB3 = 0.0
          BB4 = 0.0
          BB5 = 0.0
          DO I= 1,NPE
             AA3 = AA3 + 0.5*XJ(I) * DSKK (I)
             AA4 = AA4 +     XJ(I) * DSKE (I)
             AA5 = AA5 + 0.5*XJ(I) * DSEE (I)
             BB3 = BB3 + 0.5*YJ(I) * DSKK (I)
             BB4 = BB4 +     YJ(I) * DSKE (I)
             BB5 = BB5 + 0.5*YJ(I) * DSEE (I)
          ENDDO

C..................................................................
          ALP1 =  BB2 / DET
          ALP2 = -AA2 / DET
          BET1 = -BB1 / DET
          BET2 =  AA1 / DET
          AGG1 = -(AA3*ALP1**2+AA4*ALP1*BET1+AA5*BET1**2)
          AGG2 = -(AA3*ALP2**2+AA4*ALP2*BET2+AA5*BET2**2)
          BGG1 = -(BB3*ALP1**2+BB4*ALP1*BET1+BB5*BET1**2)
          BGG2 = -(BB3*ALP2**2+BB4*ALP2*BET2+BB5*BET2**2)
          ALP3 =   ( AGG1*BB2 - BGG1*AA2 ) /DET
          ALP5 =   ( AGG2*BB2 - BGG2*AA2 ) /DET
          BET3 =   ( BGG1*AA1 - AGG1*BB1 ) /DET
          BET5 =   ( BGG2*AA1 - AGG2*BB1 ) /DET
C..................................................................
          IF ( NCYL.EQ.3) THEN
             XR=0.0
             DO KK=1,NPE
```

```
                  XR=XR+SI(KK)*XJ(KK)
               ENDDO
               XCC=XR
            ELSE
               XCC=1.0
            ENDIF
        UM=0
        VM=0
            DO 26 M=1,NPE
               DSXM= DSIK(M)  * AJI(1,1)  + DSIE(M)  * AJI(1,2)
               DSYM= DSIK(M)  * AJI(2,1)  + DSIE(M)  * AJI(2,2)
               UMN = UN-UM
               VMN = VN-VM
               CALL UPWIND ( AKESI , ETA   , NE    , NPE   , M          ,
     1                       X     , Y     , NOD   , NELM  , NNOD       ,
     2                       AKK   , DEN   , CPP   , UMN   , VMN        ,
     3                       TAU                                        )
               WSUPG= TAU *  ((UN-UM)*DSXM+(VN-VM)*DSYM)

               DO 26 N=1,NPE
                  DSXN= DSIK(N)  * AJI(1,1)  + DSIE(N)  * AJI(1,2)
                  DSYN= DSIK(N)  * AJI(2,1)  + DSIE(N)  * AJI(2,2)

                  AKW(M,N)=AKW(M,N)+  WSUPG
     1                            *   DEN  * CPP
     2                            *   (  (UN-UM)  * DSXN  + (VN-VM)  * DSYN)
     3                            *   DET * WT(KI,IF)  * WT(KJ,IF)*XCC
                  AKWD(M,N)  = AKWD  (M,N)  + WSUPG * AKK * (
     1                         DSKK(N)*ALP1**2+
     2                         2.0*DSKE(N)*ALP1*BET1+
     3                         DSEE(N)*BET1**2+
     4                         2.0*DSIK(N)*ALP3+
     5                         2.0*DSIE(N)*BET3+
     6                         DSKK(N)*ALP2**2+
     7                         2.0*DSKE(N)*ALP2*BET2+
     8                         DSEE(N)*BET2**2+
     9                         2.0*DSIK(N)*ALP5+
     1                         2.0*DSIE(N)*BET5  )  *
     2                         DET  *  WT(KI,IF)  *  WT(KJ,IF)*XCC
  26        CONTINUE
  24    CONTINUE
C..................
C..................
 100    DO I=1,NPE
          DO J=1,NPE
            IF      ( NSUPG.EQ.1  )  THEN
                ELSTIF(I,J)= AKC(I,J)+AKV(I,J)
            ELSEIF  ( NSUPG.EQ.2  )  THEN
                ELSTIF(I,J)= AKC(I,J)+AKV(I,J)+AKW(I,J)
            ELSEIF  ( NSUPG.EQ.3  )  THEN
                ELSTIF(I,J)= AKC(I,J)+AKV(I,J)+AKW(I,J)-AKWD(I,J)
            ELSE
                STOP
            ENDIF
          ENDDO
```

```
      ENDDO
      RETURN
      END
C........................................
C     ELEMENT LOAD VECTOR FOR HEAT EQUATION    .
C........................................
      SUBROUTINE ELFTS ( NE    , NPE   , GAUSS , WT , ELF , GF   , T   ,
     1                  VPROP , IVIS  , VHS   , AK , CP  , CPHI ,
     2                  NOD   , X     , Y     , IR , IF  , NELM , NCYL,
     3                  NNOD  , NSIZ  , NSTF  , NSUPG , GFSI, NCARB )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD( NELM,9) , X(NNOD)    , Y(NNOD)             ,
     1          ELF(18)                   , GF(NSIZ)   , T(NSIZ)    ,
     2          GAUSS(7,7)    , WT(7,7)                           ,
     3          DSIK(9) , DSIE(9) , SI(9), DSIKM(9),DSIEM(9),SIM(9) ,
     4          AJI(2,2)      , AJ(2,2)    , XJ(9)      , YJ(9)      ,
     5          VPROP (30)    , VHS ( NSIZ , 5 )       , GFSI ( NSIZ )
      DIMENSION CPHI ( NSIZ )
C ........................
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C........................
      CALL ARRZRF ( ELF   , NSTF   )
C ........................
      DO 20 KI=1,IF
         AKESI=GAUSS(KI,IF)
         DO 20 KJ=1,IF
            ETA=GAUSS(KJ,IF)
C             CALL SHAPEB   ( AKESI , ETA , DSIK1 , DSIE1 , SI , NPE ,
C     1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD ,
C     2  DSIK, DSIE,UN,VN)
            CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
            CALL JACOB2  ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                   NPE , NE )
            CALL UVN     ( UN   , VN , SI,SIM, NPE   , NE     ,   GF,
     1                    NOD , NELM , NSIZ                  )
            CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
            CALL QHT ( QN    , VPROP , GF     , T    , VHS , CPHI     ,
     1                 NE    , NPE   , AKESI , ETA , NOD , X   , Y ,
     2                 NELM , NNOD  , NSIZ  , IVIS  , FVAL, NCARB )
            DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
            CPP = FVAL*CP   +(1-FVAL)*VPROP(11)
            AKK = FVAL*AK   +(1-FVAL)*VPROP(10)
            IF ( NCYL.EQ.3) THEN
               XR=0.0
               DO KK=1,NPE
                  XR=XR+SI(KK)*XJ(KK)
               ENDDO
               XCC=XR
            ELSE
               XCC=1.0
            ENDIF
```

```
          COEF= DET*WT(KI,IF)*WT(KJ,IF)*XCC
          DO 25 M=1,NPE
             IF ( NSUPG.NE.3 ) THEN
                WSUPG = 0.0
                GOTO 110
             ENDIF
             DSXM= DSIK(M)  * AJI(1,1)  + DSIE(M)  * AJI(1,2)
             DSYM= DSIK(M)  * AJI(2,1)  + DSIE(M)  * AJI(2,2)
             CALL UPWIND ( AKESI  , ETA   , NE    , NPE   , M         ,
     1                     X      , Y     , NOD   , NELM  , NNOD      ,
     2                     AKK    , DEN   , CPP   , UN    , VN        ,
     3                     TAU                                        )
             WSUPG= TAU * (UN*DSXM+VN*DSYM)
 110         ELF(M)=ELF(M)+QN*( SIM(M)+WSUPG )*COEF
 25       CONTINUE
 20   CONTINUE
      RETURN
      END
C..............................................
C     HEAT SOURCE TERM                         .
C..............................................
      SUBROUTINE QHT ( Q      , VPROP, GF , T, VHS , CPHI          ,
     1                 NE     , NPE  , AKESI , ETA , NOD , X   , Y  ,
     2                 NELM   , NNOD , NSIZ  , IVIS  , FVAL ,NCARB )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD (NELM,9) , X(NNOD) , Y(NNOD)
      DIMENSION GF  (NSIZ)   , T(NSIZ) , CPHI ( NSIZ )
      DIMENSION VPROP ( 30)  , VHS  ( NSIZ , 5 )
      CALL VISCOS ( AMU   , GAMAD , VPROP, NE    , VHS   , CPHI ,
     1              NPE   , AKESI , ETA   , GF    , T , NOD , X ,
     2              Y     , NELM  , NNOD  , NSIZ  , IVIS ,NCARB )
      AMU = (FVAL*AMU+(1-FVAL)*VPROP(8))
      Q=AMU*GAMAD**2

      RETURN
      END
C..............................................
C   ADD ELF   SEE EQ (2.111) 2ND TERM RHS      .
C..............................................
      SUBROUTINE ADDELF ( ELF , ELF1 , THETA , DT , NPE , NSTF , NDF )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ELF ( NSTF ), ELF1 (NSTF ), F ( 18 )
C............
      CALL ARRZRF ( F , NSTF )
C............
      DO I= 1,NPE*NDF
         F(I)=(  (1-THETA)*ELF1(I)  + THETA*ELF(I)  )*DT
      ENDDO
      DO I= 1,NPE*NDF
         ELF(I)=F(I)
      ENDDO

      RETURN
      END
C.......................................................
```

```
C     ELEMENT LOAD VECTOR (TEMP) SEE EQ (2.111) FIRST TERM RHS.
.
C.............................................................
      SUBROUTINE ELFT ( NE     , NPE   , DT   , THETA   ,  ELSTIF   , ELF ,
     1                  DMASS , TO    , NOD , NELM    ,
     2                  NSIZ                                      )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD     (NELM,9)                                ,
     1          TO      (NSIZ)                                  ,
     2          ELSTIF (18,18)   , ELF(18)  , DMASS(18,18)
      DO I=1,NPE
         DO J=1,NPE

            ELF(I)=ELF(I)+ ( DMASS(I,J)-DT*(1-THETA)*ELSTIF(I,J))
     1                    *    TO(NOD(NE,J))
        ENDDO
      ENDDO

      RETURN
      END
C...............................................
C     MASS MATRIX CALC. SEE EQ (3.11) NASSEHI  .
C...............................................
      SUBROUTINE MASSW ( NE     , NPE   , GAUSS , WT   , CP    ,
     1                   DMASS , NOD   , X     , Y    , IR    , IF   ,
     2                   NELM  , NNOD  , NSTF  , NSUPG , AK  , GF   ,
     3                   NSIZ  , GFSI  , VPROP , NCYL             )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD(NELM,9)      , X(NNOD)  , Y(NNOD)          ,
     1           GAUSS(7,7)       , WT(7,7)  , DMASS(18,18)     ,
     2           SI(9) , DSIE(9) , DSIK(9),DSIKM(9),DSIEM(9),SIM(9) ,
     3           XJ(9) , YJ(9)   , AJ(2,2)  , AJI(2,2)          ,
     4           GF ( NSIZ )      , GFSI ( NSIZ ) , VPROP ( 30)
      DO I= 1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C........................
      CALL ARR2ZF ( DMASS , NSTF )
C........................
      DO 24 KI= 1,IF
            AKESI=GAUSS(KI,IF)
            DO 24 KJ= 1,IF
               ETA=GAUSS(KJ,IF)
C              CALL SHAPEB  ( AKESI , ETA , DSIK1 , DSIE1 , SI , NPE ,
C     1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD ,
C     2  DSIK, DSIE,UN,VN)
                CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1      DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL JACOB2 ( AJ   , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                       NPE , NE )
               CALL UVN     ( UN   , VN , SI  ,SIM, NPE   , NE     ,  GF,
     1                       NOD , NELM , NSIZ                    )
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
               DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
```

```
                CPP = FVAL*CP   +(1-FVAL)*VPROP(11)
                AKK = FVAL*AK   +(1-FVAL)*VPROP(10)
                IF ( NCYL.EQ.3) THEN
                   XR=0.0
                   DO KK=1,NPE
                      XR=XR+SI(KK)*XJ(KK)
                   ENDDO
                   XCC=XR
                ELSE
                   XCC=1.0
                ENDIF
                DO 26 M=1,NPE
                   IF ( NSUPG.NE.3 ) THEN
                      WSUPG = 0.0
                      GOTO 110
                   ENDIF
                   DSXM= DSIK(M) * AJI(1,1) + DSIE(M) * AJI(1,2)
                   DSYM= DSIK(M) * AJI(2,1) + DSIE(M) * AJI(2,2)
                   CALL UPWIND ( AKESI , ETA  , NE   , NPE  , M       ,
     1                           X     , Y    , NOD  , NELM , NNOD    ,
     2                           AKK   , DEN  , CPP  , UN   , VN      ,
     3                           TAU                                  )
                   WSUPG= TAU * (UN*DSXM+VN*DSYM)
 110               DO 26 N=1,NPE

C-------------------------- I PUT 1 IN PLACE OF AKK*CPP. IT HAS NO MEANING
WHEN WE USE DIMLESS EQ.
C--------------------------- ENERGY EQ IS REPRESENTED IN (4.118) NASSEHI,S
BOOK ---------------

                       DMASS(M,N)= DMASS(M,N)+   1
     1                             * (SIM(M)+WSUPG) * SI(N)
     2                             * DET    * WT(KI,IF) * WT(KJ,IF)*XCC
 26             CONTINUE
 24   CONTINUE
C...
      CALL LUMP ( DMASS , NSTF , NPE )
C...
      RETURN
      END
C..............................................................
C    SOLUTION OF ENERGY EQUATION (TAYLOR-GALERKIN FORMULATION    .
C..............................................................
      SUBROUTINE TMPTGL (
     1 TF    , TFI   , GF      , TO     , SO    , THF                ,
     2 GAUSS , WT    , VHS     , GHF    , GFSO  , GFSH               ,
     3 X     , Y     , NOD     , NOP    , CPHI                       ,
     4 BCT   , NCODT , NOPPT   , MDFT   , NDNT                       ,
     5 NPE   , IF    , DT      , NDFT   , NEM                        ,
     6 NET   , NNM   , NTRAN   , NSUPG  , NCARB                      ,
     7 AK    , CP    , VPROP                                         ,
     8 NSIZ  , NSTF  , NELM    , NNOD   , MAXFR , IVIS               ,
     9 R1    , ELF   , ELSTIF  , DMASS  , ELF1                       ,
     1 LDEST , NK    , EQ      , LHED   , LPIV                       ,
     2 JMOD  , QQ    , PVKOL                                         ,
     3  NCOD   , BC     , NOPP    , MDF                                          )
C..............................
      IMPLICIT REAL*8 (A-H,O-Z)
C..............................
      DIMENSION TF     ( NSIZ ) , TFI ( NSIZ ), GF   ( NSIZ )
      DIMENSION TO     ( NSIZ ) , SO   ( NSIZ ), CPHI ( NSIZ )
      DIMENSION GFSH   ( NSIZ ) , GFSO( NSIZ )
      DIMENSION THF    ( NSIZ ) , GHF  ( NSIZ )
      DIMENSION GAUSS  ( 4,4   ) , WT( 4,4 )
      DIMENSION VPROP  ( 30    ) , VHS  ( NSIZ , 5 )
      DIMENSION X      ( NNOD ) , Y ( NNOD )
      DIMENSION NOD    ( NELM    , 9 )           , NOP ( NELM , 9)
      DIMENSION BCT    ( NSIZ ) , NCODT ( NSIZ ) , NOPPT ( NSIZ )
      DIMENSION MDFT   ( NSIZ ) , NDNT   ( NSIZ )
C............................
      DIMENSION ELF    ( NSTF )   , ELSTIF ( NSTF,NSTF )
      DIMENSION DMASS  ( NSTF       , NSTF )  , ELF1  ( NSTF )
      DIMENSION LDEST  ( NSTF   )
      DIMENSION LHED   ( MAXFR )
      DIMENSION NK     ( NSTF   )
      DIMENSION LPIV   ( MAXFR )
      DIMENSION JMOD   ( MAXFR ) , QQ      ( MAXFR )
      DIMENSION PVKOL  ( MAXFR ) , R1  ( NSIZ )
      DIMENSION EQ     ( MAXFR    , MAXFR )
      DIMENSION BC     ( NSIZ   ) , NCOD ( NSIZ ) , NOPP ( NSIZ )
      DIMENSION MDF    ( NSIZ   )
C..........................
      CALL ARRZRF ( TF , NET   )
      CALL ARRZRF ( THF, NET   )
C ....
C ....
C.....
      CALL RSAVE ( THF , TO , NSIZ )
      DO 100 K=1,2
         CALL ARRZRF ( TF , NET   )
         CALL ARRZRF ( R1 , NSIZ )
         CALL RSAVE ( BC    , BCT    , NSIZ )
         CALL RSAVI ( NCOD , NCODT , NSIZ )
         CALL RSAVI ( NOPP , NOPPT , NSIZ )
         CALL RSAVI ( MDF   , MDFT   , NSIZ )
         TMSP = DT / (3.0- DFLOAT (K) )
         DO 34 NE=1,NEM
            CALL ELFTGL  ( NE     , NPE   , GAUSS , WT     , ELF   , SO   ,
     1                     TF     , THF   , TO    , TMSP   , GHF   , K    ,
     2                     CP     , AK    , VPROP , IVIS   , VHS   ,
     3                     X      , Y     , NOD   , IR     , IF    , CPHI,
     4                     NELM   , NNOD , NSIZ   , NSTF   , ELSTIF       ,
     5                     GFSH   , GFSO , NCARB                          )
         CALL FRONT
     1( ELSTIF     , ELF     , NE     , NOP   , NELM    , NSTF , LDEST , NK      ,
     2  MAXFR      , EQ      , LHED   , LPIV  , JMOD    , QQ    , PVKOL , TF      ,
     3  R1         , NCOD    , BC     , NOPP  , MDF     , NDNT , NSIZ   , NEM     ,
     4  NSIZ       , NET     , LCOL   , NELL  , NPE                               )
 34   CONTINUE
C...................................................................
```

```
         CALL RSAVE ( THF , TF , NET )
100   CONTINUE
C....................
      RETURN
C....................
      END
C.........................................................
C  CALCULATION OF ELEMENT LOAD VECTOR FOR TAYLOR-GALERKIN MODEL .
C.........................................................
      SUBROUTINE ELFTGL  ( NE    , NPE  , GAUSS , WT    , ELF  , SO  ,
     1                     TF    , THF  , TO    , TMSP  , GHF  , K   ,
     2                     CP    , AK   , VPROP , IVIS , VHS ,
     3                     X     , Y    , NOD   , IR    , IF   , CPHI,
     4                     NELM  , NNOD , NSIZ  , NSTF  , ELSTIF     ,
     5                     GFSH  , GFSO , NCARB                      )
C...................................................
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD ( NELM,9 ) , X (NNOD) , Y(NNOD)                 ,
     1          GAUSS (7,7), WT(7,7)    , ELF (NSTF)               ,
     2          GHF    ( NSIZ ) , SO   ( NSIZ ) , ELSTIF(NSTF,NSTF)  ,
     3          TF     ( NSIZ ) , THF  ( NSIZ ) , TO ( NSIZ )        ,
     3          GFSH   ( NSIZ ) , GFSO( NSIZ )                      ,
     4          VPROP ( 30)     , VHS ( NSIZ)                       ,
     5          DSIK(9), DSIE(9) , SI(9), DSIKM(9),DSIEM(9),SIM(9) ,
     6          DSKK(9)     , DSEE(9)    , DSKE(9)                  ,
     7          XJ(9)       , YJ(9)                                 ,
     8          AJI(2,2)    , AJ(2,2)
      DIMENSION CPHI ( NSIZ )
C....................................................
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
C ...
C
      DO I=1,NSTF
         ELF(I)=0.0
         DO J=1,NSTF
            ELSTIF(I,J)=0.0
         ENDDO
      ENDDO
      DO 14 KI=1,IF
         AKESI=GAUSS(KI,IF)
         DO 14 KJ=1,IF
            ETA=GAUSS(KJ,IF)
            CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
            CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                         NPE , NE )
            IF ( K.EQ.1 ) THEN
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSO, NOD, NELM, NSIZ )
           CALL UVN  ( UN, VN, SI, SIM, NPE, NE, SO , NOD, NELM, NSIZ)
               CALL QHT  ( QN    , VPROP , SO    , TO  , VHS , CPHI   ,
     1                       NE    , NPE    , AKESI , ETA , NOD , X   , Y ,
    2                      NELM , NNOD   , NSIZ   , IVIS, FVAL, NCARB  )
          ELSE
             CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSH, NOD, NELM, NSIZ )
         CALL UVN   ( UN, VN, SI,SIM, NPE, NE, GHF, NOD, NELM, NSIZ )
             CALL QHT  ( QN    , VPROP , GHF   , THF , VHS , CPHI   ,
    1                      NE    , NPE   , AKESI , ETA , NOD , X  , Y ,
    2                      NELM , NNOD   , NSIZ   , IVIS, FVAL, NCARB  )
          ENDIF
C.....................................................................
          AKK = FVAL*AK   +(1-FVAL)*VPROP(10)
          DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
          CPP = FVAL*CP   +(1-FVAL)*VPROP(11)
C.....................................................................
          DETCOF = DET * WT(KI,IF) * WT(KJ,IF)
          TAVG = 0.0
          DTTX = 0.0
          DTTY = 0.0
          DO I=1,NPE
             DSXM= DSIK(I) * AJI(1,1) + DSIE(I) * AJI(1,2)
             DSYM= DSIK(I) * AJI(2,1) + DSIE(I) * AJI(2,2)
             TAVG = TAVG + SI(I)*TO ( NOD(NE,I) )
             DTTX = DTTX + DSXM *THF( NOD(NE,I) )
             DTTY = DTTY + DSYM *THF( NOD(NE,I) )
          ENDDO
          DO 16 M=1,NPE
             DSXM= DSIK(M) * AJI(1,1) + DSIE(M) * AJI(1,2)
             DSYM= DSIK(M) * AJI(2,1) + DSIE(M) * AJI(2,2)
             CALL UPWIND ( AKESI , ETA   , NE   , NPE  , M             ,
    1                      X     , Y     , NOD  , NELM , NNOD          ,
    2                      AKK   , DEN   , CPP  , UN   , VN            ,
    3                      TAU                                         )
             WSUPG= TAU * (UN*DSXM+VN*DSYM)
             ELF ( M ) = ELF ( M )
    1       + DEN *CPP*SI(M) * TAVG * DETCOF
    2       + (-1)*TMSP*AKK*( DSXM*DTTX+DSYM*DTTY )*DETCOF
    3       + (-1)*TMSP*DEN*CPP*(WSUPG+SI(M))*(UN*DTTX+VN*DTTY)*DETCOF
    4       + TMSP*SI(M)*QN*DETCOF
             DO 18 N=1,NPE
                ELSTIF(M,N)=ELSTIF(M,N)+DEN*CPP*SI(M)*SI(N)*DETCOF
 18          CONTINUE
 16       CONTINUE
 14   CONTINUE
      RETURN
      END
C.........................................
C     WRITE NODAL OUTPUTS                       .
C.........................................
      SUBROUTINE OUTPUT ( NNM , GF , TF , NSIZ , PSOUT , NNEE , PMG ,
    1                     CRF , PRHS     , TIME , NWR    , NTRAN       ,
    2                     ITER, IEND     , VN   , TN     , GFS         ,
    3                     NFREE , NCARB , VHS  , CPHI   , VPROP        )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION GF   ( NSIZ ) , TF  ( NSIZ ) , PSOUT ( NSIZ )
      DIMENSION NNEE ( NSIZ ) , PMG ( NSIZ ) , PRHS  ( NSIZ )
      DIMENSION CPHI ( NSIZ ) , CR  ( 10   ) , VPROP ( 30   )
      DIMENSION CRF  ( NSIZ ) , GFS ( NSIZ ) , VHS ( NSIZ , 5 )
      DIMENSION NWR (10) , VR  (10) , PS (10) , PV(10) , TR (10)
      IF  ( IEND.EQ.0) RETURN
      CALL HGSTVL ( TMAX,PMAX,TF,PRHS,PMG,NSIZ,NNM,NT,NP,NM,PMIN )
      WRITE ( 2 , 5111) ITER,VN,TN
      IF ( NTRAN.EQ.2 .OR. NTRAN.EQ.3)  WRITE ( 2 , 5115 ) TIME
      WRITE ( 2,  5112) TMAX,NT
      WRITE ( 2,  5113) PMAX,NP
      WRITE ( 2,  5114) PMIN,NM
      WRITE ( 2 , 5120 )
      DO 24  I=1,NNM
        VRES= DSQRT (GF(I+I-1)**2+GF(I+I)**2)
        PSEE= PSOUT(I) / NNEE(I)
        CALL FILTER    ( GFS(I)  , GVO )
        WRITE (2,5130) I,GF(I+I-1),GF(I+I),VRES,TF(I),PRHS(I),GVO,CRF(I)
        IF ( NFREE.NE.1 ) WRITE (30) GVO
        IF ( NCARB.NE.1 ) WRITE (31) CRF(I)
        IF ( NCARB.NE.1 ) WRITE (32) CPHI(I)
 24   CONTINUE
C..   WRITE THE STRESS COMPONENTS
      WRITE ( 2 , 5133 )
      DO I=1,NNM
         GII2 = VHS  ( I , 5 )
         GAMAD  = DSQRT ( 0.5*GII2 )
         CALL VISEQU ( AMU  , GAMAD  , TF(I) , VPROP , 2 )
         IF ( NCARB.EQ.2 ) THEN
            CPP  = CPHI (I)
            CALL BOUN01 ( CPP )
            AMU = AMU * ( VPROP(21)+VPROP(22)*CPP )
         ENDIF
         IF ( NFREE.NE.1 ) THEN
            FVAL = GFS (I)
            IF ( FVAL.LT.0.0 ) FVAL =0.0
            IF ( FVAL.GT.1.0 ) FVAL =1.0
            AMU = FVAL*AMU+(1-FVAL)*VPROP(8)
         ENDIF
         TTXX= 2 * AMU * VHS (I,1)
         TTXY= AMU * ( VHS (I,2) + VHS (I,3) )
         TTYY= 2 * AMU * VHS (I,4)
         VORT= DABS (VHS (I,3)-VHS(I,2))
         IF ((GAMAD+VORT).NE.0.0) ALAM= GAMAD / (GAMAD+VORT)
         WRITE ( 2 , 5130 ) I,TTXX,TTXY,TTYY,AMU,GAMAD,VORT,ALAM
      ENDDO
C..   WRITING OF OUTPUT RESULTS FOR TRANSIENT SOLUTION
      IF (NTRAN.EQ.2 .OR. NTRAN.EQ.3 .OR. NTRAN.EQ.4 ) THEN
         DO K=1,3
            I      = NWR(K)
            VR (K) = DSQRT (GF(I+I-1)**2+GF(I+I)**2)
            PS (K) = PSOUT(I) / NNEE(I)
            PV (K) = PRHS (I)
            TR (K) = TF (I)
            CR (K) = CPHI(I)
         ENDDO
         WRITE ( 4 , 5125 ) TIME , (VR(I),PV(I),TR(I),CR(I),I=1,3)
      ENDIF
```

```
C.......................................................................
 5111 FORMAT (1X,/,'-  SOLUTION AFTER',I5,'  ITERATION(S) -',
     1        1X,/,'-  ERROR OVAL ( FL** ) =',F20.9,
     2        1X,/,'-  ERROR OVAL ( TP** ) =',F20.9          )
 5112 FORMAT (1X,'  MAXIMUM TEMPERATURE = ',G20.5,'  AT NODE =',I5)
 5113 FORMAT (1X,'  MAXIMUM PRESSURE    = ',G20.5,'  AT NODE =',I5)
 5114 FORMAT (1X,'  MINIMUM PRESSURE    = ',G20.5,'  AT NODE =',I5)
 5115 FORMAT (1X,'SOLUTION AT TIME = ',G20.5,/)
 5120 FORMAT (1X,//,' RESULT ( NODE NO. ,VX, VY, |V|, TEMPERATURE, PRESS
     1URE, FREE SURFACE FUNCTION, CONCENTRATION)',/)
 5125 FORMAT (1X,E11.6,3(' |',4E12.4))
 5130 FORMAT (1X,I4,2X,7(D11.5,2X))
 5133 FORMAT (1X,//,1X,'TXX, TXY, TYY, AMU, GAMMAD, VORT, LAMBDA ',/)
      RETURN
      END
C........................................
C     MASS MATRIX CALC.                        .
C........................................
      SUBROUTINE MASST ( NE    , NPE  , GAUSS , WT  , CP   , DNS ,
     1                   DMASS , NOD  , X     , Y   , IR   , IF  ,
     2                   NELM  , NNOD , NSTF                       )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD(NELM,9)     , X(NNOD) , Y(NNOD)         ,
     1           GAUSS(7,7)      , WT(7,7) , DMASS(18,18)    ,
     2           SI(9) , DSIE(9) , DSIK(9) ,  DSIKM(9),DSIEM(9),SIM(9) ,
     3           XJ(9) , YJ(9)   , AJ(2,2) , AJI(2,2)
      DO I= 1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO

C.......................
      CALL ARR2ZF ( DMASS , NSTF )
C.....................
      DO 24 KI= 1,IF
            AKESI=GAUSS(KI,IF)
            DO 24 KJ= 1,IF
               ETA=GAUSS(KJ,IF)

               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1       DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)



               CALL JACOB2 ( AJ  , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                       NPE , NE )


               DO 26 M=1,NPE
                  DO 26 N=1,NPE
                      DMASS(M,N)= DMASS(M,N)+  CP*DNS
     1                            * SIM(M) * SIM(N)
     2                            * DET    * WT(KI,IF) * WT(KJ,IF)
 26            CONTINUE
 24   CONTINUE
```

```
C......................
      CALL LUMP ( DMASS , NSTF , NPE )
C......................
      RETURN
      END
C....................................................
C    CALCULATION OF PRESSURE                        .
C....................................................
      SUBROUTINE PRESS ( NEM    , GAUSS , NPE   , GF    , NOD  , NCYL    ,
     1                   X      , Y     , IR    , IF    , NELM , NNOD    ,
     2                   NSIZ   , VPROP , T     , VHS   , CPHI , NCARB   ,
     3                   PSOUT  , NNM   , IVIS  , PRHS  , WT   , GFS     ,
     4                   PMG                                             )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD (NELM,9)  , X(NNOD)  , Y(NNOD)   ,
     1          GF   (NSIZ)    , T ( NSIZ )           ,
     2          GAUSS(7,7)     , WT (7,7)             ,
     3          DSIK(9)     , DSIE(9) , SI(9), DSIKM(9),DSIEM(9),SIM(9) ,
     4          XJ(9)          , YJ(9)                ,
     5          AJ(2,2)        , AJI(2,2)             ,
     6          COE (9,5)      , XX (4)               ,
     7          PSOUT ( NSIZ )                        ,
     8          PRHS  ( NNOD ) , PMG ( NSIZ     )     ,
     9          VPROP ( 30   ) , VHS ( NSIZ , 5)      ,
     1          GFS   ( NSIZ ) , CPHI ( NSIZ )
C...........................
      CALL ARRZRF ( PSOUT , NSIZ )
      CALL ARRZRF ( PRHS  , NNOD )
      DO 40  NE=1,NEM
         NID=0
         DO I=1,NPE
            XJ(I)=X(NOD(NE,I))
            YJ(I)=Y(NOD(NE,I))
         ENDDO
         DO 70  II=1,IR
            DO 70  JJ=1,IR
               NID=NID+1
               AKESI=GAUSS(II,IR)
               ETA=GAUSS(JJ,IR)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1       DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL JACOB2 ( AJ   , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                       NPE , NE )
               CALL VISCOS ( AMU   , GAMAD , VPROP, NE    , VHS , CPHI  ,
     1                       NPE   , AKESI , ETA   , GF    , T , NOD , X ,
     2                       Y     , NELM  , NNOD , NSIZ , IVIS , NCARB)
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFS , NOD, NELM, NSIZ )
               CALL UVN   ( UN    , VN , SI,SIM , NPE   , NE , GF , NOD ,
     1                    NELM , NSIZ            )
               AMU = FVAL*AMU+(1-FVAL)*VPROP(8)
               GUX=0.0
               GVY=0.0
               XR =0.0
               DO 48 I=1,NPE
```

```
                  DFX=GF(2*NOD(NE,I)-1)
                  DFY=GF(2*NOD(NE,I))
                  RUX=(DSIK(I)*AJI(1,1)+DSIE(I)*AJI(1,2))
                  RVY=(DSIK(I)*AJI(2,1)+DSIE(I)*AJI(2,2))
                  GUX=GUX+DFX*RUX
                  GVY=GVY+DFY*RVY
                  XR=XR+SI(I)*XJ(I)
 48            CONTINUE
               IF (NCYL.EQ.1 )  PRESSE=-VPROP(15)*AMU*( GUX + GVY )
               IF (NCYL.EQ.3 )  PRESSE=-VPROP(15)*AMU*( GUX+UN/XR+GVY )
C...........................................................
               DO I= 1,NPE
                  PRHS ( NOD (NE,I) ) = PRHS ( NOD (NE,I) ) + PRESSE*
     1            SIM(I)*DET*WT(II,IR)*WT(JJ,IR)
               ENDDO
C...........................................................
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , 4   ,
     1       DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
               COE (NID,1)=SI(1)
               COE (NID,2)=SI(2)
               COE (NID,3)=SI(3)
               COE (NID,4)=SI(4)
               COE (NID,5)=PRESSE
70          CONTINUE
            CALL GJE (COE,XX,4)
            DO I = 1,4
               PSOUT ( NOD(NE,I) ) = PSOUT ( NOD ( NE,I ) ) + XX(I)
            ENDDO
40       CONTINUE
C...........................................................
         DO I=1,NNM
            PRHS (I) = PRHS (I) / PMG (I)
         ENDDO
         RETURN
         END
C............................................................
C        GLOBAL MATRIX CALCULATION FOR VARIATIONAL  RECOVERY METHOD .
C............................................................
         SUBROUTINE PVRGMX ( NNM    , NEM    , NPE    , NOD , NNOD , NELM
     1                             , PMG    , IR     , GAUSS , WT   , X    , Y
     2                             , NSTF   , DMASS , IF     , NSIZ            )
         IMPLICIT REAL*8 (A-H,O-Z)
         DIMENSION  NOD(NELM,9)      , X(NNOD) , Y(NNOD)          ,
     1              GAUSS(7,7)       , WT(7,7) , DMASS(18,18)     ,
     2              PMG ( NSIZ )
C.......................
         CALL ARR2ZF ( DMASS , NSTF )
         CALL ARRZRF ( PMG    , NNOD )
C.......................
         DO NE= 1,NEM

               CALL MASST  ( NE     , NPE    , GAUSS , WT , 1.0D0, 1.0D0 ,
     1                       DMASS , NOD    , X      , Y  , IR    , IF ,
     2                       NELM   , NNOD , NSTF                        )
```

```
C........................
         DO I= 1,NPE
            PMG ( NOD (NE,I )) = PMG ( NOD (NE,I) ) + DMASS(I,I)
         ENDDO

      ENDDO

      RETURN
      END
C.......................................................
C  FIND CONNECTIVITY OF EACH NODE ADJ. TO A GIVEN ELEMENT  .
C.......................................................
      SUBROUTINE  ANODAE ( NNM , NEM , NPE , NOD , NNEE , NNOD , NELM )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD (NELM,9) , NNEE  ( NNOD )
C..............................................
      CALL ARRZRI ( NNEE , NNOD )
      DO I= 1,NNM
         DO J= 1,NEM
            DO K= 1,NPE
               IF ( NOD (J,K) .EQ. I ) NNEE (I)=NNEE(I)+1
            ENDDO
         ENDDO
      ENDDO
      RETURN
      END
C.............................................................
C     SUBROUTINE FOR THE CULCULATION OF VELOCITY COMPONENT        .
C     GRADIENT  USING VARIATIONAL RECOVERY                        .
C     FORMULATION                                                 .
C.............................................................
      SUBROUTINE  VISRHD ( VHS    , GFI    , X     , Y     , NOD  , IR    ,
     1                     IF     , GAUSS  , WT    , NPE   , NELM , NNOD  ,
     2                     NSIZ   , NSTF   , PMG   , NEM   , NNM          )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD (NELM,9)  , X(NNOD)  , Y(NNOD)
      DIMENSION GFI (NSIZ)    , PMG ( NSIZ )
      DIMENSION GAUSS(7,7)    , WT  (7,7)
      DIMENSION DSIK(9)       , DSIE(9)  , SI(9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION XJ(9)         , YJ(9)
      DIMENSION AJ(2,2)       , AJI(2,2)
      DIMENSION VHS  ( NSIZ , 5 )
C............................................
      INDEX = IR
C............................................
      DO I= 1, NSIZ
         DO J= 1,4
            VHS  (I,J)=0.0
         ENDDO
      ENDDO
C............................................
      DO 100 NE= 1,NEM
         DO I=1,NPE
            XJ(I)=X(NOD(NE,I))
            YJ(I)=Y(NOD(NE,I))
```

```
       ENDDO
       DO 70  II=1,INDEX
          DO 70  JJ=1,INDEX
             AKESI= GAUSS(II,INDEX)
             ETA  = GAUSS(JJ,INDEX)
             CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1         DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
             CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                     NPE , NE )
             GUX = 0.0
             GUY = 0.0
             GVX = 0.0
             GVY = 0.0
             DO 48 I=1,NPE
                DFX=GFI ( 2*NOD(NE,I)-1 )
                DFY=GFI ( 2*NOD(NE,I)   )
                RUX=DFX*(DSIK(I)*AJI(1,1)+DSIE(I)*AJI(1,2))
                RUY=DFX*(DSIK(I)*AJI(2,1)+DSIE(I)*AJI(2,2))
                RVX=DFY*(DSIK(I)*AJI(1,1)+DSIE(I)*AJI(1,2))
                RVY=DFY*(DSIK(I)*AJI(2,1)+DSIE(I)*AJI(2,2))
                GUX=GUX+RUX
                GUY=GUY+RUY
                GVX=GVX+RVX
                GVY=GVY+RVY
 48          CONTINUE
             DO M= 1,NPE
                DETCOE = DET*WT(II,INDEX)*WT(JJ,INDEX)
                VHS ( NOD ( NE,M )  , 1 ) = VHS ( NOD ( NE,M ) , 1 ) +
     1                                      GUX * SIM(M)* DETCOE
                VHS ( NOD ( NE,M )  , 2 ) = VHS ( NOD ( NE,M ) , 2 ) +
     1                                      GUY * SIM(M)* DETCOE
                VHS ( NOD ( NE,M )  , 3 ) = VHS ( NOD ( NE,M ) , 3 ) +
     1                                      GVX * SIM(M)* DETCOE
                VHS ( NOD ( NE,M )  , 4 ) = VHS ( NOD ( NE,M ) , 4 ) +
     1                                      GVY * SIM(M)* DETCOE
             ENDDO
 70    CONTINUE
 100  CONTINUE
      DO I= 1,NNM
         DO J= 1,4
            VHS ( I , J ) = VHS ( I , J ) / PMG (I)
         ENDDO
         VHS ( I , 5 )   = (  2* VHS ( I , 1 ) )**2 +
     1                     (  2* VHS ( I , 4 ) )**2 +
     2                     2 * ( VHS ( I , 2 ) + VHS ( I , 3 ) )**2
      ENDDO
      RETURN
      END
C.............................................
C     CALCULATION OF MAXIMUM TEMPERATURE         .
C     AND PRESSURE                               .
C.............................................
      SUBROUTINE HGSTVL ( TMAX, PMAX , TF   , PRHS , PMG , NSIZ,
     1                    NNM , NT   , NP   , NM   , PMIN      )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION TF   ( NSIZ )
      DIMENSION PMG ( NSIZ ) , PRHS  ( NSIZ )
      TMAX= TF(1)
      PMAX= PRHS (1)
      PMIN= PRHS (1)
      NT=1
      NP=1
      NM=1
      DO I=2,NNM
         TM= TF(I)
         PM= PRHS (I)
         PI= PRHS (I)
         IF ( TM.GT.TMAX ) THEN
            TMAX=TM
            NT  =I
         ENDIF
         IF ( PM.GT.PMAX ) THEN
            PMAX=PM
            NP  =I
         ENDIF
         IF ( PI.LT.PMIN ) THEN
            PMIN = PI
            NM   = I
         ENDIF
      ENDDO
      RETURN
      END
C.........................................................
C                                                         .
C     MESH UPDATING FOR PURE LAGRANGIAN FORMULATION       .
C                                                         .
C.........................................................
      SUBROUTINE MSHUPD ( X , Y , GF , NNOD , NSIZ , DT , NNM )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X ( NNOD ),Y( NNOD ), GF( NSIZ )
      DO I=1,NNM
         X(I)=X(I)*GF(I+I-1)*DT
         Y(I)=Y(I)*GF(I+I  )*DT
      ENDDO
      RETURN
      END
C...............................................................
C     PREPARATION OF THE INITIAL DATA FOR  MOVING MESH MODEL    .
C     USING INTERPOLATION METHOD                                .
C...............................................................
C
      SUBROUTINE  MSHINI ( X    , Y   , NNM  , NNOD  , NELM  , NPE    ,
     1                     NSIZ , CORP, NODPV, AINIT , SO    , TO     ,
     2                     GFSO , CRO , CPHIO, NCARB , NFREE , NNISO  ,
     3                     DT                                         )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(NNOD) , Y(NNOD)
      DIMENSION CORP  ( NNOD , 2 ) , NODPV ( NELM , 9 )
      DIMENSION AINIT ( NSIZ , 5 ) , XJ(9) , YJ(9)
      DIMENSION SO ( NSIZ ) , TO ( NSIZ ) , GFSO ( NSIZ ) , CRO ( NSIZ )
      DIMENSION CPHIO ( NSIZ )
      DIMENSION SI(9) , DSIE(9) , DSIK(9),DSIKM(9),DSIEM(9),SIM(9)
C...............................................................
      REWIND 12
      READ  ( 12 ) NNMP,NEMP
      DO I=1,NEMP
         READ ( 12 ) (NODPV(I,J),J=1,NPE )
      ENDDO
C...............................................................
      DO I=1,NNMP
         READ (20) AINIT(I+I-1,1)     , AINIT(I+I,1)  , AINIT(I,2)  ,
     1             AINIT(I     ,3 )   , AINIT(I   ,4)  , AINIT(I,5)  ,
     2             CORP ( I,1 ) , CORP (I,2)
         ENDDO
C
C
C
      DO K=1,NNM
         DO NE=1,NEMP
            DELTAX= 2.0/ 20.0
            DELTAY= 2.0/ 20.0
            DO I=1,NPE
               XJ (I)=CORP ( NODPV(NE,I) , 1 )
               YJ (I)=CORP ( NODPV(NE,I) , 2 )
            ENDDO
            DO I=1,21
               ETA  = -1.0+ DELTAY * (I-1)
               DO J=1,21
                  AKESI= -1.0 + DELTAX * ( J-1 )
                  CALL COORD (AKESI,ETA,NPE,XJ,YJ,XCR,YCR )
                  CALL SHAPE (AKESI,ETA,DSIK,DSIE,SI,NPE,
     1      DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
                  VXX =0.0
                  VYY =0.0
                  TEM =0.0
                  FFS =0.0
                  CBB =0.0
                  CPP =0.0
                  DO KK=1,NPE
                     VXX =VXX+AINIT (2*NODPV(NE,KK)-1,1)*SI(KK)
                     VYY =VYY+AINIT (2*NODPV(NE,KK)   ,1)*SI(KK)
                     IF ( NNISO.EQ.2 )
     1                  TEM =TEM+AINIT (NODPV(NE,KK),2)*SI(KK)
                     IF ( NFREE.NE.1 )
     1                  FFS =FFS+AINIT (NODPV(NE,KK),3)*SI(KK)
                     IF ( NCARB.EQ.2 )
     1                  CBB =CBB+AINIT (NODPV(NE,KK),4)*SI(KK)
                     IF ( NCARB.EQ.2 )
     1                  CPP =CPP+AINIT (NODPV(NE,KK),5)*SI(KK)
                  ENDDO
                  XCRN = XCR + DT * VXX
                  YCRN = YCR + DT * VYY
                  IF ( DABS( (X(K)-XCRN)/XCRN ).LE.0.0001.AND.
     1                DABS( (Y(K)-YCRN)/YCRN ).LE.0.0001    ) THEN
                      SO   ( K+K-1) = VXX
```

```
                            SO    ( K+K   ) = VYY
                            TO    ( K )     = TEM
                            GFSO  ( K )     = FFS
                            CRO   ( K )     = CBB
                            CPHIO( K )      = CPP
                            GOTO 100
                       ENDIF
                    ENDDO
                 ENDDO
              ENDDO
 100       ENDDO
           RETURN
           END
C..............................................
C       A=0 SUBROUTINE   ( INTEGER )          .
C..............................................
      SUBROUTINE ARRZRI ( IA , N )
      DIMENSION IA(N)
      DO I= 1,N
         IA(I)=0
      ENDDO
      RETURN
      END
C..............................................
C       A=0 SUBROUTINE   ( FLOAT    )         .
C..............................................
      SUBROUTINE ARRZRF  ( A , N )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N)
      DO I= 1,N
         A(I)=0.0
      ENDDO
      RETURN
      END
C..............................................
C       A=0 SUBROUTINE   ( 2-D ) ( FLOAT    )  .
C..............................................
      SUBROUTINE ARR2ZF ( A , N )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N)
      DO I= 1,N
         DO J=1,N
            A( I,J )=0.0
         ENDDO
      ENDDO
      RETURN
      END
C..............................................
C     A=B SUBROUTINE   ( FLOAT )              .
C..............................................
      SUBROUTINE RSAVE ( A , B , N )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N),B(N)
      DO I=1,N
         A(I)=B(I)
      ENDDO
      RETURN
      END
C...............................................
C     A=B SUBROUTINE   ( INTEGER )            .
C...............................................
      SUBROUTINE RSAVI ( IA , IB , N )
      DIMENSION IA(N),IB(N)
      DO I=1,N
         IA(I)=IB(I)
      ENDDO
      RETURN
      END

C.........................................................
C      CALCULATION OF THE ELEMENT LENGTH FOR THE BILINEAR.
C      ELEMENT                                          .
C.........................................................
      SUBROUTINE ELMTLNTH ( NE , NOD  , X   , Y ,NELM, NNOD, ELLGTH,
      1                        DSIX, DSIY)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD ( NELM,9 ) , X (NNOD) , Y(NNOD)
      DIMENSION  DSI(2) , NP(2,4), NQ(4,2)
C.......................................................................
      NQ(1,2)=2
      NQ(1,1)=1
      NQ(2,2)=3
      NQ(2,1)=2
      NQ(3,2)=4
      NQ(3,1)=3
      NQ(4,2)=1
      NQ(4,1)=4
      DO I=1,4
         DO J=1,2
            NP (3-J,I) = NOD ( NE , NQ (I,3-J) )
         ENDDO
      ENDDO

            DSIX =0.0
            DSIY =0.0
            DSI(1)=0.5
            DSI(2)=0.5
            DO L=1,4
              DO I= 1,2
               DSIX = (X(NP(1,2))-X(NP(1,1))+X(NP(1,3))-X(NP(2,3)))/2
               DSIY = (Y(NP(2,2))-Y(NP(1,2))+Y(NP(1,4))-Y(NP(2,4)))/2
             ENDDO
            ENDDO

C
C     CALCULATION OF ELEMENT LENGTH AND THE COMPONENTS OF THE
C     UNIT VECTOR NORMAL TO THE BOUNDARY
C
C
```

```
             ELLGTH = DSQRT(DSIX*DSIY)

      RETURN
      END


      SUBROUTINE SHAPEB ( AKESI , ETA , DSIK1 , DSIE1 , SI , NPE ,
      1 DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD ,
     2 DSIK, DSIE,UN,VN )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DSIK1(9),DSIE1(9),SI(9),DSIKM(9),DSIEM(9),SIM(9),X(NNOD)
      DIMENSION DSIK(9),DSIE(9)


      CALL  ELMTLNTH ( NE , NOD  , X    , Y ,NELM, NNOD, ELLGTH,
      1                      DSIX, DSIY)



C============== high order in time nothing in x,   B4, B6, B8, ARE FOR
ACTIVATE DIFFERENT ORDERS =========================
C===============FOR TEST WE ADD BUBBLE FOR TIME BY SUM INSTEAD OF
MULTIPLICATION==================================



         B= 0.03
         BT=0

         DSIK1(1)=-0.25*(1-ETA)-2*B*AKESI


         DSIK1(2)= 0.25*(1-ETA)+2*B*AKESI


         DSIK1(3)= 0.25*(1+ETA)+2*B*AKESI


         DSIK1(4)=-0.25*(1+ETA)-2*B*AKESI

C................................


         DSIE1(1)=-0.25*(1-AKESI)-2*BT*ETA-BT*4*ETA*(1-ETA**3)

         DSIE1(2)=-0.25*(1+AKESI)-2*BT*ETA-BT*4*ETA*(1-ETA**3)

         DSIE1(3)= 0.25*(1+AKESI)+2*BT*ETA+BT*4*ETA*(1-ETA**3)

         DSIE1(4)= 0.25*(1-AKESI)+2*BT*ETA+BT*4*ETA*(1-ETA**3)

         DSIK(1)=-0.25*(1-ETA)
         DSIK(2)= 0.25*(1-ETA)
      DSIK(3)= 0.25*(1+ETA)
      DSIK(4)=-0.25*(1+ETA)

C................................

      DSIE(1)=-0.25*(1-AKESI)
      DSIE(2)=-0.25*(1+AKESI)
      DSIE(3)= 0.25*(1+AKESI)
      DSIE(4)= 0.25*(1-AKESI)

C.................................

       SI(1)=0.25*(1-AKESI)*(1-ETA)+B*(1-AKESI**2)+BT*(1-ETA**2)+
    1          BT*(1-ETA**4)

       SI(2)=0.25*(1+AKESI)*(1-ETA)-B*(1-AKESI**2)+BT*(1-ETA**2)+
    1          BT*(1-ETA**4)

       SI(3)=0.25*(1+AKESI)*(1+ETA)+B*(1-AKESI**2)+BT*(1-ETA**2)+
    1          BT*(1-ETA**4)

       SI(4)=0.25*(1-AKESI)*(1+ETA)-B*(1-AKESI**2)+BT*(1-ETA**2)+
    1          BT*(1-ETA**4)


C=====================================================================
===========================================




c-----------------wieght functions

      DSIKM(1)=-0.25*(1-ETA)
      DSIKM(2)= 0.25*(1-ETA)
      DSIKM(3)= 0.25*(1+ETA)
      DSIKM(4)=-0.25*(1+ETA)
C.................................
      DSIEM(1)=-0.25*(1-AKESI)
      DSIEM(2)=-0.25*(1+AKESI)
      DSIEM(3)= 0.25*(1+AKESI)
      DSIEM(4)= 0.25*(1-AKESI)
C.................................
      SIM(1)=0.25*(1-AKESI)*(1-ETA)
      SIM(2)=0.25*(1+AKESI)*(1-ETA)
      SIM(3)=0.25*(1+AKESI)*(1+ETA)
      SIM(4)=0.25*(1-AKESI)*(1+ETA)

    RETURN
    END
C.............................................................
C     CALCULATION OF SHAPE FUNCTIONS AND THEIR DERIVATIVES
```

```
C......................................................
      SUBROUTINE SHAPESH ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1 DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD,
     1                 AMU   , GAMAD , VPROP,  VHS, CPHI         ,
     1            G  , T  ,NSIZ    , IVIS,NCARB)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DSIK(9),DSIE(9),SI(9),DSIKM(9),DSIEM(9),SIM(9),X(NNOD)


      CALL ELMTLNTH ( NE , NOD , X   , Y ,NELM, NNOD , ELLGTH,
      1                  DSIX, DSIY )

           Da=0.0001
         GAMAD=GAMAD**(-0.05)
C        GAMAD=GAMAD/1000
c        IF (GAMAD.LT.0.0001) GAMAD=0.0001
C        PRINT*,GAMAD
         B=5/(8*(1+(10*Da)/(ELLGTH**2)*GAMAD))
         B=0.0

        DSIK(1)=-0.25*(1-ETA)
        DSIK(2)= 0.25*(1-ETA)
        DSIK(3)= 0.25*(1+ETA)
        DSIK(4)=-0.25*(1+ETA)
C..............................
        DSIE(1)=-0.25*(1-AKESI)
        DSIE(2)=-0.25*(1+AKESI)
        DSIE(3)= 0.25*(1+AKESI)
        DSIE(4)= 0.25*(1-AKESI)
C...............................
        SI(1)=0.25*(1-AKESI)*(1-ETA)-B*(1-ETA**2)*(1-AKESI**2)
        SI(2)=0.25*(1+AKESI)*(1-ETA)-B*(1-ETA**2)*(1-AKESI**2)
        SI(3)=0.25*(1+AKESI)*(1+ETA)-B*(1-ETA**2)*(1-AKESI**2)
        SI(4)=0.25*(1-AKESI)*(1+ETA)-B*(1-ETA**2)*(1-AKESI**2)

        DSIKM(1)=-0.25*(1-ETA)
        DSIKM(2)= 0.25*(1-ETA)
        DSIKM(3)= 0.25*(1+ETA)
        DSIKM(4)=-0.25*(1+ETA)
C...............................
        DSIEM(1)=-0.25*(1-AKESI)
        DSIEM(2)=-0.25*(1+AKESI)
        DSIEM(3)= 0.25*(1+AKESI)
        DSIEM(4)= 0.25*(1-AKESI)
C...............................
        SIM(1)=0.25*(1-AKESI)*(1-ETA)
        SIM(2)=0.25*(1+AKESI)*(1-ETA)
        SIM(3)=0.25*(1+AKESI)*(1+ETA)
        SIM(4)=0.25*(1-AKESI)*(1+ETA)

      RETURN
      END
      SUBROUTINE SHAPE ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1 DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)


      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DSIK(9),DSIE(9),SI(9),DSIKM(9),DSIEM(9),SIM(9),X(NNOD)


            CALL ELMTLNTH ( NE , NOD , X    , Y ,NELM, NNOD , ELLGTH,
     1                                DSIX, DSIY)


      IF (NPE.EQ.4) THEN




         DSIK(1)=-0.25*(1-ETA)
         DSIK(2)= 0.25*(1-ETA)
         DSIK(3)= 0.25*(1+ETA)
         DSIK(4)=-0.25*(1+ETA)
C................................
         DSIE(1)=-0.25*(1-AKESI)
         DSIE(2)=-0.25*(1+AKESI)
         DSIE(3)= 0.25*(1+AKESI)
         DSIE(4)= 0.25*(1-AKESI)
C.................................
         SI(1)=0.25*(1-AKESI)*(1-ETA)
         SI(2)=0.25*(1+AKESI)*(1-ETA)
         SI(3)=0.25*(1+AKESI)*(1+ETA)
         SI(4)=0.25*(1-AKESI)*(1+ETA)



c--------wieght functions

         DSIKM(1)=-0.25*(1-ETA)
         DSIKM(2)= 0.25*(1-ETA)
         DSIKM(3)= 0.25*(1+ETA)
         DSIKM(4)=-0.25*(1+ETA)
C.................................
         DSIEM(1)=-0.25*(1-AKESI)
         DSIEM(2)=-0.25*(1+AKESI)
         DSIEM(3)= 0.25*(1+AKESI)
         DSIEM(4)= 0.25*(1-AKESI)
C................................
         SIM(1)=0.25*(1-AKESI)*(1-ETA)
         SIM(2)=0.25*(1+AKESI)*(1-ETA)
         SIM(3)=0.25*(1+AKESI)*(1+ETA)
         SIM(4)=0.25*(1-AKESI)*(1+ETA)

      ELSEIF (NPE.EQ.8) THEN
```

```
         DSIK(1)=0.5*AKESI-0.5*AKESI*ETA-0.25*ETA**2+0.25*ETA
         DSIK(2)=0.5*AKESI-0.5*AKESI*ETA+0.25*ETA**2-0.25*ETA
         DSIK(3)=0.5*AKESI+0.5*AKESI*ETA+0.25*ETA**2+0.25*ETA
         DSIK(4)=0.5*AKESI+0.5*AKESI*ETA-0.25*ETA**2-0.25*ETA
         DSIK(5)=AKESI*(-1+ETA)
         DSIK(6)=0.5-0.5*ETA**2
         DSIK(7)=-AKESI*(1+ETA)
         DSIK(8)=-0.5+0.5*ETA**2
C...............................
         DSIE(1)=0.5*ETA-0.25*AKESI**2-0.5*AKESI*ETA+0.25*AKESI
         DSIE(2)=0.5*ETA-0.25*AKESI**2+0.5*ETA*AKESI-0.25*AKESI
         DSIE(3)=0.5*ETA+0.25*AKESI**2+0.5*AKESI*ETA+0.25*AKESI
         DSIE(4)=0.5*ETA+0.25*AKESI**2-0.5*AKESI*ETA-0.25*AKESI
         DSIE(5)=-0.5+0.5*AKESI**2
         DSIE(6)=-(1+AKESI)*ETA
         DSIE(7)=0.5-0.5*AKESI**2
         DSIE(8)=(-1+AKESI)*ETA
C...............................
         SI(1)=0.25*(1-AKESI)*(1-ETA)*(-1-AKESI-ETA)
         SI(2)=0.25*(1+AKESI)*(1-ETA)*(-1+AKESI-ETA)
         SI(3)=0.25*(1+AKESI)*(1+ETA)*(-1+AKESI+ETA)
         SI(4)=0.25*(1-AKESI)*(1+ETA)*(-1-AKESI+ETA)
         SI(5)=0.5*(1-AKESI**2)*(1-ETA)
         SI(6)=0.5*(1+AKESI)*(1-ETA**2)
         SI(7)=0.5*(1-AKESI**2)*(1+ETA)
         SI(8)=0.5*(1-AKESI)*(1-ETA**2)
      ELSEIF (NPE.EQ.9) THEN
         DSIK(1)=0.25*(2*AKESI-1)*(ETA**2-ETA)
         DSIK(2)=0.25*(2*AKESI+1)*(ETA**2-ETA)
         DSIK(3)=0.25*(2*AKESI+1)*(ETA**2+ETA)
         DSIK(4)=0.25*(2*AKESI-1)*(ETA**2+ETA)
         DSIK(5)=-AKESI*(ETA**2-ETA)
         DSIK(6)=0.5*(2*AKESI+1)*(1-ETA**2)
         DSIK(7)=-AKESI*(ETA**2+ETA)
         DSIK(8)=0.5*(2*AKESI-1)*(1-ETA**2)
         DSIK(9)=-2*AKESI*(1-ETA**2)
C...............................
         DSIE(1)=0.25*(AKESI**2-AKESI)*(2*ETA-1)
         DSIE(2)=0.25*(AKESI**2+AKESI)*(2*ETA-1)
         DSIE(3)=0.25*(AKESI**2+AKESI)*(2*ETA+1)
         DSIE(4)=0.25*(AKESI**2-AKESI)*(2*ETA+1)
         DSIE(5)=0.5*(1-AKESI**2)*(2*ETA-1)
         DSIE(6)=-ETA*(AKESI**2+AKESI)
         DSIE(7)=0.5*(1-AKESI**2)*(2*ETA+1)
         DSIE(8)=-ETA*(AKESI**2-AKESI)
         DSIE(9)=-2*ETA*(1-AKESI**2)
C...............................
         SI(1)=0.25*(AKESI**2-AKESI)*(ETA**2-ETA)
         SI(2)=0.25*(AKESI**2+AKESI)*(ETA**2-ETA)
         SI(3)=0.25*(AKESI**2+AKESI)*(ETA**2+ETA)
         SI(4)=0.25*(AKESI**2-AKESI)*(ETA**2+ETA)
         SI(5)=0.5*(1-AKESI**2)*(ETA**2-ETA)
         SI(6)=0.5*(AKESI**2+AKESI)*(1-ETA**2)
         SI(7)=0.5*(1-AKESI**2)*(ETA**2+ETA)
         SI(8)=0.5*(AKESI**2-AKESI)*(1-ETA**2)
         SI(9)=(1-AKESI**2)*(1-ETA**2)
      ENDIF
      RETURN
      END
C.......................................................
C  CALCULATION OF JACOBIAN
C.......................................................
      SUBROUTINE JACOB2 ( AJ   , AJI   , DET , X , Y ,
     1                   DSIK , DSIE , N    , NE )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AJ (2,2) , AJI (2,2) ,
     1          X(N)      , Y(N)       ,
     2          DSIK(N)   , DSIE(N)

      DO I=1,2
         DO J=1,2
            AJ  (I,J)  = 0.0
            AJI (I,J)  = 0.0
         ENDDO
      ENDDO
      DO I=1,N
         AJ(1,1) = AJ(1,1) + X(I) * DSIK(I)
         AJ(1,2) = AJ(1,2) + Y(I) * DSIK(I)
         AJ(2,1) = AJ(2,1) + X(I) * DSIE(I)
         AJ(2,2) = AJ(2,2) + Y(I) * DSIE(I)
      ENDDO
      DET= AJ(1,1)*AJ(2,2)-AJ(1,2)*AJ(2,1)

      IF (DET.LE.0.0) THEN
         WRITE (2,110) NE,DET
 110     FORMAT  (1X,'ERROR : ZERRO OR NEGATIVE JACOBIAN=',I6,G20.5)
         STOP
      ENDIF

      AJI(1,1) =  AJ(2,2) / DET
      AJI(1,2) = -AJ(1,2) / DET
      AJI(2,1) = -AJ(2,1) / DET
      AJI(2,2) =  AJ(1,1) / DET

      RETURN
      END
C..................................................
C     TRANSPOSE OF  A MATRIX                     .
C..................................................
      SUBROUTINE TRAP(A,N)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(9,9)
      DO 10 I=1,N
         DO 10 J=1,N
            IF (I.GE.J) GOTO 10
            TEMP=A(I,J)
            A(I,J)=A(J,I)
            A(J,I)=TEMP
 10         CONTINUE
      RETURN
      END
C.....................................................
C    CALCULATION OF ERROR NORM                       .
C.....................................................
      SUBROUTINE VNORM ( VN , NEQ , G1 , G2 , NSIZ )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION G2(NSIZ),G1(NSIZ)
      A=0.0
      B=0.0
      DO I=1,NEQ
         A=A+(G2(I)-G1(I))**2
         B=B+G2(I)**2
      ENDDO
      IF ( A.LT.1.00D-10.AND.B.LT.1.00D-10 ) THEN
         VN =0.0
      ELSE
         VN=DSQRT(A)/DSQRT(B)
      ENDIF
      RETURN
      END
C............................................
C     MASS LUMPING                          .
C............................................
      SUBROUTINE LUMP ( DMASS , NSTF , N   )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  DMASS ( NSTF , NSTF )
C.........................
      DO I=1,N
         AA=0.0
         DO K=1,N
            AA=AA+DMASS(I,K)
         ENDDO
         DMASS(I,I)=AA
      ENDDO
      DO I=1,N
         DO J=1,N
            IF ( I.NE.J ) DMASS(I,J)=0.0
         ENDDO
      ENDDO
C...
      RETURN
      END
C...........................................
C     GAUSS JORDAN ELIMINATION WITH PIVOTING  .
C...........................................
      SUBROUTINE GJE (A,X,N)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(4,5),X(4)
      M=N+1
      N1=N-1
      DO 6 K=1,N
      K1=K+1
      K2=K
C.............
      B0=DABS(A(K,K))
      DO 1 I=K,N
      B1=DABS (A(I,K))
      IF ((B0-B1).LT.0.0) THEN
      B0=B1
      K2=I
      ENDIF
 1    CONTINUE
C............
      IF ((K2-K).NE.0) THEN
C............
      DO 2 J=K,M
      C=A(K2,J)
      A(K2,J)=A(K,J)
 2    A(K,J)=C
      ENDIF
C...........
 3    DO 4 J=K1,M
 4    A(K,J)=A(K,J)/A(K,K)
      A(K,K)=1.0
      DO 6 I=1,N
      IF (I.NE.K) THEN
      DO 5  J=K1,M
 5    A(I,J)=A(I,J)-A(I,K)*A(K,J)
      A(I,K)=0.0
      ENDIF
 6    CONTINUE
C
      DO 7 I=1,N
 7    X(I)=A(I,M)
      RETURN
      END
C.........................................
C     1D SHAPE FUNCTIONS CALCULATION          .
C.........................................
      SUBROUTINE LAGSH1 ( AKESI , SI , DSI )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION SI(3),DSI(3)
        SI(1)   = -0.5*AKESI*(1.0-AKESI)
        SI(2)   =  (1.0+AKESI)*(1.0-AKESI)
        SI(3)   =  0.5*AKESI*(1.0+AKESI)
        DSI(1)  = -0.5+AKESI
        DSI(2)  = -2.0*AKESI
        DSI(3)  =  0.5+AKESI
      RETURN
      END
C...........................................................
C  CALCULATION OF COMPONENTS OF UNIT VECTOR NORMAL TO BOUNDARY   .
C...........................................................
      SUBROUTINE UTNML ( M , X , Y , NP, DNX , DNY , ELLGTH  , NNOD ,
     1                    NSDOF    , NSIZ  , NW                     )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X ( NNOD )          , Y( NNOD )
      DIMENSION NSDOF ( NSIZ , 3 )
      DIMENSION DSI(3) , SI(3)   , NP(3)
C..............................................
```

```
      IF ( M.EQ.NSDOF ( NW , 1 ) ) THEN
         AKESI = -1.0
      ELSEIF ( M.EQ.NSDOF ( NW , 2 ) ) THEN
        AKESI =  0.0
      ELSEIF ( M.EQ.NSDOF ( NW , 3 ) ) THEN
        AKESI = +1.0
      ELSE
        RETURN
      ENDIF
      CALL LAGSH1 ( AKESI , SI , DSI )
      DSIX =0.0
      DSIY =0.0
      DO I= 1,3
         DSIX = DSIX + DSI(I)*X( NP(I) )
         DSIY = DSIY + DSI(I)*Y( NP(I) )
      ENDDO
      ELLGTH = DSQRT ( DSIX **2 + DSIY  **2 )
      DCELL  =  DSIX  / ELLGTH
      DCELM  =  DSIY  / ELLGTH
      DNX    =  DCELM
      DNY    = -DCELL
      ELLGTH = DSQRT ( (X(NP(3))-X(NP(1)))**2 + (Y(NP(3))-Y(NP(1)))**2)
      RETURN
      END
C..............................................
C     RENEWAL SUBROUTINE  WITH OVER-RELAXATION .
C..............................................
      SUBROUTINE RENWAL ( A , B , N , OMEGA )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N),B(N)
      DO I=1,N
         A(I)=A(I)+(B(I)-A(I))*OMEGA
      ENDDO
      RETURN
      END
C...............................................
C     BOUNDS THE VALUE OF A BETWEEN 0 AND 1     .
C...............................................
      SUBROUTINE BOUN01 ( A )
      IMPLICIT REAL*8 (A-H,O-Z)
      IF ( A.LT.0.0 ) A=0.0
      IF ( A.GT.1.0 ) A=1.0
      RETURN
      END
C.............................................
C     CALCULATION OF COORDINATES             .
C.............................................
      SUBROUTINE COORD ( AKESI , ETA , NPE , XJ , YJ, X ,Y )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XJ(9)   , YJ(9)   ,
     1          DSIK(9) , DSIE(9) , SI(9),DSIKM(9),DSIEM(9),SIM(9)
      X=0.0
      Y=0.0
      CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1   DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
      DO I=1,NPE
         X=X+XJ(I)*SI(I)
         Y=Y+YJ(I)*SI(I)
      ENDDO
      RETURN
      END
C.........................................
C    CALCULATION OF UNIT VECTORS COMPONENTS   .
C.........................................
      SUBROUTINE UNITVC (AKESI, ETA , NE , NPE   , UK1 , UK2 , UE1 , UE2,
     1                   NOD  , X   , Y  , NELM , NNOD                  )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD  ( NELM , 9 ) , X    ( NNOD  ) , Y  ( NNOD )
      DIMENSION DSIK (9), DSIE (9), SI (9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION XJ   ( 9          ) , YJ   ( 9     )
      DIMENSION AJI  ( 2    , 2 ) , AJ   ( 2 , 2 )
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
      CALL SHAPE  ( AKESI, ETA , DSIK , DSIE , SI , NPE,
      1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
      CALL JACOB2 ( AJ    , AJI , DET   , XJ    , YJ , DSIK , DSIE ,
     1              NPE   , NE                                     )
      UK1=AJ(1,1)
      UK2=AJ(1,2)
      UE1=AJ(2,1)
      UE2=AJ(2,2)
      DK=UK1**2+UK2**2
      DE=UE1**2+UE2**2
      UK1=UK1/DSQRT(DK)
      UK2=UK2/DSQRT(DK)
      UE1=UE1/DSQRT(DE)
      UE2=UE2/DSQRT(DE)
      RETURN
      END
C.........................................
C     CALCULATION OF TERMS IN UPWIND FORMULA .
C.........................................
      SUBROUTINE COEFF ( PECLET , ALPHA )
      IMPLICIT REAL*8 (A-H,O-Z)
      IF ( DABS ( PECLET) .LT. 1.0D-05 ) THEN
         ALPHA=0.0
      ELSEIF (PECLET .GE.  20 ) THEN
         ALPHA=  1.0-1.0/PECLET
      ELSEIF (PECLET .LE. -20 ) THEN
         ALPHA= -1.0-1.0/PECLET
      ELSE
         ALPHA= ( 1.0 / DTANH  ( PECLET ) ) - 1.0 / PECLET
      ENDIF
      RETURN
      END
C.........................................
C     CALCULATION OF TERMS IN UPWIND FORMULA .
C.........................................
```

```
      SUBROUTINE COBET (PECLET,BETA)
      IMPLICIT REAL*8 (A-H,O-Z)
      IF ( DABS (PECLET) .LT.  1.0D-05) THEN
         BETA=0.0
      ELSEIF (PECLET .GT . 300 )  THEN
         BETA=1.0
      ELSEIF (PECLET .LT .-300 )  THEN
         BETA=-1.0
      ELSE
        SH= DSINH ( 2*PECLET )
        CH= DCOSH ( 2*PECLET )
        TH= DTANH (   PECLET )
        PINV = 1.0 / PECLET
        P2   = 2   * PECLET
        BETA=(2-CH-2*PINV*TH+(1./P2)*SH)/(4*TH-SH-3*PINV*SH*TH)
      ENDIF
      RETURN
      END
C........................................
C     CALCULATION OF UPWIND PARAMETERS       .
C........................................
      SUBROUTINE UWPARA (BETK,BETE,ALFK,ALFE,DIAK,DIET,ALAMK,ALAME,M)
      IMPLICIT REAL*8 (A-H,O-Z)
      AKC=BETK*DIAK
      AKM=0.5*ALFK*DIAK
      AEC=BETE*DIET
      AEM=0.5*ALFE*DIET
      IF (M.EQ.1) THEN
         ALAMK=AKC
         ALAME=AEC
      ENDIF
      IF (M.EQ.2) THEN
         ALAMK=AKC
         ALAME=AEC
      ENDIF
      IF (M.EQ.3) THEN
         ALAMK=AKC
         ALAME=AEC
      ENDIF
      IF (M.EQ.4) THEN
         ALAMK=AKC
         ALAME=AEC
      ENDIF
      IF (M.EQ.5) THEN
         ALAMK=AKM
         ALAME=AEC
      ENDIF
      IF (M.EQ.6) THEN
         ALAMK=AKC
         ALAME=AEM
      ENDIF
      IF (M.EQ.7) THEN
         ALAMK=AKM
         ALAME=AEC
      ENDIF
      IF (M.EQ.8) THEN
         ALAMK=AKC
         ALAME=AEM
      ENDIF
      IF (M.EQ.9) THEN
         ALAMK=AKM
         ALAME=AEM
      ENDIF
      RETURN
      END
C...........................................................
C     CALCULATION OF UPWINDING MULRIPLIER                   .
C...........................................................
      SUBROUTINE UPWIND ( AKESI , ETA   , NE   , NPE  , M      ,
     1                    X     , Y     , NOD  , NELM , NNOD   ,
     2                    AK    , DNS   , CP   , UN   , VN     ,
     3                    TAU                                  )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD ( NELM,9 ) , X (NNOD) , Y(NNOD)             ,
     1          DSIK(9) , DSIE(9) , SI(9),DSIKM(9),DSIEM(9),SIM(9),
     2          XJ(9)       , YJ(9)                             ,
     3          AJI(2,2)    , AJ(2,2)
C.......................................................
      DIFFUS = AK / ( DNS*CP )
      DIFFUS = 0.01

C.......................................................
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
C ..
C
      CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
      CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1            NPE , NE )
      CALL UNITVC (AKESI , ETA, NE , NPE  , UK1 , UK2, UE1, UE2,
     1            NOD   , X  , Y  , NELM , NNOD                )
C.......

      VNORM=DSQRT ( UN**2 + VN**2 )

C.......
      UAK=UN*UK1+VN*UK2
      UET=UN*UE1+VN*UE2

      IF ( NPE.EQ.4 ) MDLE = 2
      IF ( NPE.EQ.8.OR.NPE.EQ.9 ) MDLE= 1
      DDX=MDLE * DABS ( AJ ( 1,1 ) + AJ ( 2,1 ) )
      DDY=MDLE * DABS ( AJ ( 1,2 ) + AJ ( 2,2 ) )
      DDK= UK1 * DDX + UK2 * DDY
      DDE= UE1 * DDX + UE2 * DDY
C
```

```
C....  PECLET NUMBER CALCULATIONS ..................
C
      IF ( DIFFUS.GE.1.0D-05 ) THEN
c           PEK=(UAK*DDK)/(2* DIFFUS )
c           PEE=(UET*DDE)/(2* DIFFUS )
c--------------dimensionless equation----------------
          PEK=(DDK)/(2* DIFFUS )
          PEE=(DDE)/(2* DIFFUS )
      ELSE
          PEK = 1.0D10
          PEE = 1.0D10
      ENDIF
C.....................................................
      CALL COEFF (PEK,ALFK)
      CALL COEFF (PEE,ALFE)
      CALL COBET (PEK,BETK)
      CALL COBET (PEE,BETE)
C.....
      CALL UWPARA ( BETK  , BETE  , ALFK , ALFE , DDK , DDE ,
     1              ALAMK , ALAME , M                       )
      TAU= DABS ( ALAMK ) * ( UAK**2 ) +  DABS ( ALAME ) * ( UET**2 )
      IF ( VNORM .LT.  1.00D-10 ) THEN
           TAU =0.0
      ELSE
           TAU= (TAU)  / VNORM  **3

C         TAU= (ALFK*DDK*UAK+ALFE*DDE*UET)/(2*VNORM**2)


       TAU= 0.01*(DDX*UN+DDY*VN)/(2*VNORM**2)


      ENDIF
      RETURN
      END
C.......................................................
C    CALCULATION OF HIGHER ORDER DERIVATIVES OF        .
C    THE SHAPE FUNCTIONS FOR 9-NODE ELEMENTS           .
C.......................................................
      SUBROUTINE SH9DD  ( AKESI , ETA , DSKK , DSEE,  DSKE     )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  DSKK(9) , DSEE(9) , DSKE(9)
C.......................................................
      DSKK( 1 ) =  0.5 *(ETA**2-ETA)
      DSKK( 2 ) =  0.5 *(ETA**2-ETA)
      DSKK( 3 ) =  0.5 *(ETA**2+ETA)
      DSKK( 4 ) =  0.5 *(ETA**2+ETA)
      DSKK( 5 ) = -1.0 *(ETA**2-ETA)
      DSKK( 6 ) =  1.0 *(1-ETA**2)
      DSKK( 7 ) = -1.0 *(ETA**2+ETA)
      DSKK( 8 ) =  1.0 *(1-ETA**2)
      DSKK( 9 ) = -2.0 *(1-ETA**2)
C.......................................................
      DSEE( 1 ) =  0.5 *(AKESI**2-AKESI)
      DSEE( 2 ) =  0.5 *(AKESI**2+AKESI)
      DSEE( 3 ) =  0.5 *(AKESI**2+AKESI)
      DSEE( 4 ) =  0.5 *(AKESI**2-AKESI)
      DSEE( 5 ) =  1.0 *(1-AKESI**2)
      DSEE( 6 ) = -1.0 *(AKESI**2+AKESI)
      DSEE( 7 ) =  1.0 *(1-AKESI**2)
      DSEE( 8 ) = -1.0 *(AKESI**2-AKESI)
      DSEE( 9 ) = -2.0 *(1-AKESI**2)
C.....................................................
      DSKE( 1 ) =  0.5 *(2*ETA-1.0)
      DSKE( 2 ) =  0.5 *(2*ETA-1.0)
      DSKE( 3 ) =  0.5 *(2*ETA+1.0)
      DSKE( 4 ) =  0.5 *(2*ETA+1.0)
      DSKE( 5 ) = -1.0 *(2*ETA-1.0)
      DSKE( 6 ) =  1.0 *(-2.0*ETA)
      DSKE( 7 ) = -1.0 *( 2.0*ETA+1.0)
      DSKE( 8 ) =  1.0 *(-2.0*ETA)
      DSKE( 9 ) = -2.0 *(-2.0*ETA)
C.....................................................
      RETURN
      END

C.....................................................
C    CALCULATION OF HIGHER ORDER DERIVATIVES OF         .
C    THE SHAPE FUNCTIONS FOR 9-NODE ELEMENTS            .
C.....................................................
      SUBROUTINE SH4DD  ( AKESI , ETA , DSKK , DSEE,  DSKE      )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION   DSKE(4), DSEE(4), DSKK(4)
      DSKE( 1 ) = 0.25
      DSKE( 2 ) = 0.25
      DSKE( 3 ) = 0.25
      DSKE( 4 ) = 0.25
      DSKK( 1 ) = 0
      DSKK( 2 ) = 0
      DSKK( 3 ) = 0
      DSKK( 4 ) = 0
      DSEE( 1 ) = 0
      DSEE( 2 ) = 0
      DSEE( 3 ) = 0
      DSEE( 4 ) = 0
      RETURN
      END

C...................................................
C    SUBROUTINE FOR CALCULATION OF THE VALUE F        .
C...................................................
      SUBROUTINE FRSFVL (
     1 GFS    , GFSI  , GFSO    , GF     , GFM                     ,
     2 GAUSS  , WT                                                  ,
     3 X      , Y     , NOD     , NOP                               ,
     4 BCF    , NCODF , NOPPF   , MDFF   , NDNF                     ,
     5 NPE    , IR    , IF      , DT     , THETA , NDFF , NEM       ,
     6 NEF    , NNM                                                 ,
     8 NSIZ   , NSTF  , NELM    , NNOD   , MAXFR , IVIS             ,
     9 R1     , ELF   , ELSTIF  , DMASS                             ,
     1  LDEST , NK     , EQ      , LHED   , LPIV                                       ,
     2  JMOD  , QQ     , PVKOL                                                         ,
     3  NCOD  , BC     , NOPP    , MDF                                                 )
C...............................
      IMPLICIT REAL*8 (A-H,O-Z)
C...............................
      DIMENSION GFS    ( NSIZ ) , GFSI ( NSIZ ), GFSO   ( NSIZ )
      DIMENSION GF     ( NSIZ ) , GFM ( NSIZ )
      DIMENSION GAUSS  ( 7,7  ) , WT( 7,7 )
      DIMENSION X      ( NNOD ) , Y ( NNOD )
      DIMENSION NOD    ( NELM   , 9 )              , NOP ( NELM , 9)
      DIMENSION BCF    ( NSIZ ) , NCODF ( NSIZ ) , NOPPF ( NSIZ )
      DIMENSION MDFF   ( NSIZ ) , NDNF  ( NSIZ )
C.............................
      DIMENSION ELF    ( NSTF )   , ELSTIF ( NSTF,NSTF )
      DIMENSION DMASS  ( NSTF     , NSTF )
      DIMENSION LDEST  ( NSTF   )
      DIMENSION LHED   ( MAXFR )
      DIMENSION NK     ( NSTF   )
      DIMENSION LPIV   ( MAXFR )
      DIMENSION JMOD   ( MAXFR ) , QQ     ( MAXFR )
      DIMENSION PVKOL  ( MAXFR ) , R1  ( NSIZ )
      DIMENSION EQ     ( MAXFR   , MAXFR )
      DIMENSION BC     ( NSIZ   ) , NCOD ( NSIZ ) , NOPP ( NSIZ )
      DIMENSION MDF    ( NSIZ   )
C.....................................................
      DIMENSION  XJ   (9)     , YJ   (9)
      DIMENSION  DSIK (9), DSIE (9), SI(9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION  AJ   (2,2) , AJI  (2,2)
C..........................
      CALL ARRZRF ( GFS  , NEF  )
      CALL ARRZRF ( R1   , NSIZ )
C ....
C ....
      CALL RSAVE ( BC    , BCF    , NSIZ )
      CALL RSAVI ( NCOD  , NCODF  , NSIZ )
      CALL RSAVI ( NOPP  , NOPPF  , NSIZ )
      CALL RSAVI ( MDF   , MDFF   , NSIZ )
C.....
      DO 34 NE=1,NEM
         CALL ARR2ZF  ( ELSTIF , NSTF )
         CALL ARR2ZF  ( DMASS  , NSTF )
         CALL ARRZRF  ( ELF    , NSTF )
         DO I=1,NPE
            XJ(I)=X(NOD(NE,I))
            YJ(I)=Y(NOD(NE,I))
         ENDDO
         DO 14 KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO 14 KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE,
     1   DSIKM , DSIEM , SIM ,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                          NPE , NE )
```

```
            CALL UVN     ( UN   , VN , SI,SIM, NPE   , NE     ,   GF,
     1                     NOD , NELM , NSIZ                     )
            CALL UVN   ( UM    , VM , SI,SIM , NPE   , NE , GFM, NOD ,
     1                   NELM , NSIZ             )
            UN = UN-UM
            VN = VN-VM
            DO 16 M=1,NPE
               DSXM= DSIK(M) * AJI(1,1) + DSIE(M) * AJI(1,2)
               DSYM= DSIK(M) * AJI(2,1) + DSIE(M) * AJI(2,2)
               CALL UPWIND ( AKESI , ETA   , NE    , NPE   , M              ,
     1                       X     , Y     , NOD   , NELM  , NNOD           ,
     2                       0.0D0 , 1.0D0, 1.0D0, UN    , VN               ,
     3                       TAU                                            )
               WSUPG= TAU * (UN*DSXM+VN*DSYM)
               DO 16 N=1,NPE
                  DSXN= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                  DSYN= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)
                  ELSTIF (M,N) = ELSTIF (M,N) + (WSUPG+SI(M) )*
     1                                ( UN   * DSXN + VN * DSYN )
     2                              *   DET * WT(KI,IF) * WT(KJ,IF)
                  DMASS (M,N) = DMASS (M,N) + SI(M)*SI(N)
     1                              *   DET * WT(KI,IF) * WT(KJ,IF)
 16            CONTINUE
 14       CONTINUE
          CALL ELFT    ( NE     , NPE   , DT   , THETA , ELSTIF , ELF   ,
     1                   DMASS , GFSO , NOD , NELM ,  NSIZ             )
          CALL ADDSF   ( NPE,DT,THETA,ELSTIF,DMASS,NDFF)
          CALL FRONT
     1( ELSTIF    , ELF    , NE    , NOP  , NELM    , NSTF , LDEST , NK     ,
     2  MAXFR     , EQ     , LHED , LPIV , JMOD    , QQ   , PVKOL , GFS    ,
     3  R1        , NCOD   , BC    , NOPP , MDF     , NDNF , NSIZ  , NEM    ,
     4  NSIZ      , NEF    , LCOL , NELL , NPE                             )
  34  CONTINUE
C....................
      RETURN
      END
C.......................................................
C     CALCULATION OF (F) IN INTERIOR ELEMENTS            .
C.......................................................
      SUBROUTINE FPSL ( FVAL ,SI ,NPE  ,NE, G   , NOD, NELM, NSIZ )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD   (NELM,9)
      DIMENSION  G     (NSIZ)
      DIMENSION  SI    (9)
      FVAL=0.0
      DO I=1,NPE
         FVAL = FVAL + SI(I) *G ( NOD(NE,I))
      ENDDO
      IF ( FVAL.LT.0.0 ) FVAL =0.0
      IF ( FVAL.GT.1.0 ) FVAL =1.0
      RETURN
      END
C.....................
C     FILTER
C.....................
```

```
      SUBROUTINE FILTER ( GVI , GVO )
      IMPLICIT REAL*8 (A-H,O-Z)
      IF ( GVI.LE.0.3 ) GVO =0.0
      IF ( GVI.GE.0.7 ) GVO =1.0
      IF ( GVI.GT.0.3.AND.GVI.LT.0.7 ) GVO =0.5
      RETURN
      END
C.................................................
C     CALCULATION OF CONCENTRATION               .
C.................................................
      SUBROUTINE CARBON (
     1 CRF    , GF     , CRO     , GFS    , GFM                          ,
     2 GAUSS  , WT                                                       ,
     3 X      , Y      , NOD     , NOP                                   ,
     4 BCC    , NCODC  , NOPPC   , MDFC   , NDNC                         ,
     5 NPE    , IR     , IF      , DT     , THETA , NDFC , NEM           ,
     6 NEC    , NNM                                                      ,
     7 NSIZ   , NSTF   , NELM    , NNOD   , MAXFR                        ,
     8 R1     , ELF    , ELSTIF  , DMASS                                 ,
     9 LDEST  , NK     , EQ      , LHED   , LPIV                         ,
     1 JMOD   , QQ     , PVKOL                                           ,
     2 NCOD   , BC     , NOPP    , MDF                                   )
C.................................
      IMPLICIT REAL*8 (A-H,O-Z)
C.................................
      DIMENSION CRF   ( NSIZ ) , GF ( NSIZ ) , CRO ( NSIZ )
      DIMENSION GFS   ( NSIZ ) , GFM ( NSIZ)
      DIMENSION GAUSS ( 7,7  ) , WT( 7,7 )
      DIMENSION X     ( NNOD ) , Y ( NNOD )
      DIMENSION NOD   ( NELM   , 9 )             , NOP ( NELM , 9)
      DIMENSION BCC   ( NSIZ ) , NCODC ( NSIZ ) , NOPPC ( NSIZ )
      DIMENSION MDFC  ( NSIZ ) , NDNC  ( NSIZ )
C.............................
      DIMENSION ELF   ( NSTF )  , ELSTIF ( NSTF,NSTF )
      DIMENSION DMASS ( NSTF , NSTF )
      DIMENSION LDEST ( NSTF  )
      DIMENSION LHED  ( MAXFR )
      DIMENSION NK    ( NSTF  )
      DIMENSION LPIV  ( MAXFR )
      DIMENSION JMOD  ( MAXFR ) , QQ    ( MAXFR )
      DIMENSION PVKOL ( MAXFR ) , R1 ( NSIZ )
      DIMENSION EQ    ( MAXFR   , MAXFR )
      DIMENSION BC    ( NSIZ  ) , NCOD ( NSIZ ) , NOPP ( NSIZ )
      DIMENSION MDF   ( NSIZ  )
C........................................................
      DIMENSION  XJ  (9)    , YJ   (9)
      DIMENSION  DSIK (9), DSIE (9), SI (9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION  AJ   (2,2) , AJI  (2,2)
C.........................
      CALL ARRZRF ( CRF  , NEC  )
      CALL ARRZRF ( R1   , NSIZ )
      CALL ARRZRF ( BC   , NSIZ )
      CALL ARRZRI ( NCOD , NSIZ )
C
C..   MODIFICATION OF BOUNDARY CONDITION TO REMOVE THE VOID AREA FROM
```

```
C..   THE SOLUTION DOMAIN
C
      DO I=1,NNM
         IF ( GFS(I).LT.0.3 ) THEN
            NCOD (I) = 1
            BC   (I) = 0.0
         ELSE
            NCOD (I) = NCODC (I)
            BC   (I) = BCC   (I)
         ENDIF
      ENDDO
C ....
C ....
      CALL RSAVI ( NOPP , NOPPC , NSIZ )
      CALL RSAVI ( MDF  , MDFC  , NSIZ )
C.....
      DO 34 NE=1,NEM
         CALL ARR2ZF ( ELSTIF , NSTF )
         CALL ARR2ZF ( DMASS  , NSTF )
         CALL ARRZRF ( ELF    , NSTF )
         DO I=1,NPE
            XJ(I)=X(NOD(NE,I))
            YJ(I)=Y(NOD(NE,I))
         ENDDO
         DO 14 KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO 14 KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1 DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
    1                        NPE , NE )
               CALL UVN     ( UN   , VN , SI ,SIM, NPE   , NE   , GF   ,
    1                         NOD , NELM , NSIZ                    )
               CALL UVN ( UM    , VM , SI,SIM, NPE   , NE , GFM, NOD ,
    1                       NELM , NSIZ              )
               UN = UN - UM
               VN = VN - VM
               DO 16 M=1,NPE
                  DSXM= DSIK(M) * AJI(1,1) + DSIE(M) * AJI(1,2)
                  DSYM= DSIK(M) * AJI(2,1) + DSIE(M) * AJI(2,2)
                  CALL UPWIND ( AKESI , ETA  , NE   , NPE  , M            ,
    1                           X     , Y    , NOD  , NELM , NNOD         ,
    2                           0.0D0 , 1.0D0, 1.0D0, UN   , VN           ,
    3                           TAU                                       )
                  WSUPG= TAU * (UN*DSXM+VN*DSYM)
                  DO 16 N=1,NPE
                     DSXN= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                     DSYN= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)
                     ELSTIF (M,N) = ELSTIF (M,N) + (WSUPG+SI(M) )*
    1                                ( UN  * DSXN + VN * DSYN )
    2                               *  DET * WT(KI,IF) * WT(KJ,IF)
                     DMASS (M,N) = DMASS (M,N) + SI(M)*SI(N)
    1                               *  DET * WT(KI,IF) * WT(KJ,IF)
 16               CONTINUE
 14      CONTINUE
         CALL ELFT    ( NE     , NPE   , DT   , THETA , ELSTIF , ELF   ,
    1                   DMASS  , CRO   , NOD  , NELM  ,  NSIZ           )
         CALL ADDSF   ( NPE,DT,THETA,ELSTIF,DMASS,NDFC)
         CALL FRONT
    1( ELSTIF    , ELF    , NE    , NOP   , NELM    , NSTF  , LDEST , NK     ,
    2  MAXFR     , EQ     , LHED  , LPIV  , JMOD    , QQ    , PVKOL , CRF    ,
    3  R1        , NCOD   , BC    , NOPP  , MDF     , NDNC  , NSIZ  , NEM    ,
    4  NSIZ      , NEC    , LCOL  , NELL  , NPE                              )
  34  CONTINUE
      RETURN
      END
C...........................................................
C     CALCULATION OF EFFECTIVE FILLER VOLUME FRACTION       .
C...........................................................
      SUBROUTINE EFVF    (
    1  CPHI  , GF     , CPHIO   , CPHIA , CRF    , CRO  , GFM              ,
    2  GAUSS , WT     , VPROP   , GFS                                       ,
    3  X     , Y      , NOD     , NOP                                       ,
    4  BCC   , NCODC  , NOPPC   , MDFC  , NDNC                              ,
    5  NPE   , IR     , IF      , DT    , THETA  , NDFC , NEM               ,
    6  NEC   , NNM                                                          ,
    7  NSIZ  , NSTF   , NELM    , NNOD  , MAXFR                             ,
    8  R1    , ELF    , ELSTIF  , DMASS                                     ,
    9  LDEST , NK     , EQ      , LHED  , LPIV                              ,
    1  JMOD  , QQ     , PVKOL                                               ,
    2  NCOD  , BC     , NOPP    , MDF                                       )
C.............................
      IMPLICIT REAL*8 (A-H,O-Z)
C.............................
      DIMENSION CPHI  ( NSIZ ) , GF ( NSIZ ) , CPHIO ( NSIZ )
      DIMENSION CPHIA ( NSIZ ) , CRF ( NSIZ ), CRO   ( NSIZ )
      DIMENSION GFM   ( NSIZ )
      DIMENSION GAUSS ( 7,7  ) , WT( 7,7 )   , VPROP (30)
      DIMENSION X     ( NNOD ) , Y ( NNOD )  , GFS ( NSIZ )
      DIMENSION NOD   ( NELM   , 9 )            , NOP ( NELM , 9)
      DIMENSION BCC   ( NSIZ ) , NCODC ( NSIZ ) , NOPPC ( NSIZ )
      DIMENSION MDFC  ( NSIZ ) , NDNC  ( NSIZ )
C..........................
      DIMENSION ELF   ( NSTF )   , ELSTIF ( NSTF,NSTF )
      DIMENSION ELF1  ( 18   )   , ELF2   ( 18   )
      DIMENSION DMASS ( NSTF , NSTF )
      DIMENSION LDEST ( NSTF   )
      DIMENSION LHED  ( MAXFR )
      DIMENSION NK    ( NSTF   )
      DIMENSION LPIV  ( MAXFR )
      DIMENSION JMOD  ( MAXFR ) , QQ     ( MAXFR )
      DIMENSION PVKOL ( MAXFR ) , R1 ( NSIZ )
      DIMENSION EQ    ( MAXFR   , MAXFR )
      DIMENSION BC    ( NSIZ   ) , NCOD ( NSIZ ) , NOPP ( NSIZ )
      DIMENSION MDF   ( NSIZ   )
C......................................................
      DIMENSION  XJ  (9)     , YJ   (9)
      DIMENSION  DSIK (9), DSIE (9), SI  (9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION  AJ   (2,2)  , AJI  (2,2)
```

```
C.........................
C.. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
      CHTIME = VPROP(7)
C.. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
      CALL ARRZRF ( CPHI , NEC  )
      CALL ARRZRF ( R1   , NSIZ )
      CALL ARRZRF ( BC   , NSIZ )
      CALL ARRZRI ( NCOD , NSIZ )
C
C..   MODIFICATION OF BOUNDARY CONDITION TO REMOVE THE VOID AREA FROM
C..   THE SOLUTION DOMAIN
C
      DO I=1,NNM
         IF ( GFS(I).LT.0.3 ) THEN
            NCOD (I) = 1
            BC   (I) = 0.0
         ELSE
            NCOD (I) = NCODC (I)
            BC   (I) = BCC   (I)
         ENDIF
      ENDDO
C ....
C ....
      CALL RSAVI ( NOPP , NOPPC , NSIZ )
      CALL RSAVI ( MDF  , MDFC  , NSIZ )
C.....
      DO 34 NE=1,NEM
         CALL ARR2ZF ( ELSTIF , NSTF )
         CALL ARR2ZF ( DMASS  , NSTF )
         CALL ARRZRF ( ELF    , NSTF )
         CALL ARRZRF ( ELF1   , NSTF )
         CALL ARRZRF ( ELF2   , NSTF )
         DO I=1,NPE
            XJ(I)=X(NOD(NE,I))
            YJ(I)=Y(NOD(NE,I))
         ENDDO
         DO 14 KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO 14 KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1        DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                       NPE , NE )
               CALL UVN    ( UN   , VN , SI,SIM , NPE  , NE  , GF   ,
     1                         NOD , NELM , NSIZ                  )
               CALL UVN  ( UM   , VM , SI,SIM, NPE , NE , GFM, NOD ,
     1                     NELM , NSIZ             )
               UN = UN - UM
               VN = VN - VM
               CBBI=0.0
               CPPI=0.0
               CBB =0.0
               CPP =0.0
               DO I=1,NPE
```

```
                   CBBI=CBBI+CRO    ( NOD(NE,I) )*SI(I)
                   CPPI=CPPI+CPHIO  ( NOD(NE,I) )*SI(I)
                   CBB =CBB +CRF    ( NOD(NE,I) )*SI(I)
                   CPP =CPP +CPHIA  ( NOD(NE,I) )*SI(I)
                ENDDO
                CALL BOUN01 ( CBBI )
                CALL BOUN01 ( CPPI )
                CALL BOUN01 ( CBB  )
                CALL BOUN01 ( CPP  )
                DO 26 M=1,NPE
                   DSXM= DSIKM(M) * AJI(1,1) + DSIEM(M) * AJI(1,2)
                   DSYM= DSIKM(M) * AJI(2,1) + DSIEM(M) * AJI(2,2)
                   CALL UPWIND ( AKESI , ETA  , NE   , NPE  , M        ,
     1                           X     , Y    , NOD  , NELM , NNOD     ,
     2                           0.0D0 , 1.0D0, 1.0D0, UN   , VN       ,
     3                           TAU                                   )
                   WSUPG= TAU * (UN*DSXM+VN*DSYM)
                   DO 16 N=1,NPE
                      DSXN= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                      DSYN= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)
                      ELSTIF (M,N) = ELSTIF (M,N) + (WSUPG+SI(M) )*
     1                               ( UN  * DSXN + VN * DSYN )
     2                              *  DET * WT(KI,IF) * WT(KJ,IF)
                      DMASS (M,N) = DMASS (M,N) + SI(M)*SI(N)
     1                              *  DET * WT(KI,IF) * WT(KJ,IF)
16                 CONTINUE
                   ELF1(M)=ELF1(M)+SI(M)*(DABS ( CBBI- CPPI )/CHTIME )
     1                          *  DET * WT(KI,IF) * WT(KJ,IF)
                   ELF2(M)=ELF2(M)+SI(M)*(DABS ( CBB - CPP  )/CHTIME )
     1                          *  DET * WT(KI,IF) * WT(KJ,IF)
                   IF ( GFS(NOD(NE,M)).LT.0.3 ) THEN
                      ELF1(M)= 0.0
                      ELF2(M)= 0.0
                   ENDIF
                   CCBB = CRO    ( NOD(NE,M ))
                   EEFF = CPHIO  ( NOD(NE,M ))
                   CALL BOUN01  ( EEFF )
                   CALL BOUN01  ( CCBB )
                   IF (EEFF.EQ.0.0.OR.CCBB.EQ.0.0 ) THEN
                      OC = 0.0
                   ELSE
                      OC= DABS ((EEFF-CCBB)/(EEFF*CCBB+EEFF-CCBB))
                   ENDIF
                   IF ( OC.GE.0.67 )  THEN
                      ELF1(M)= 0.0
                      ELF2(M)= 0.0
                   ENDIF
26              CONTINUE
14       CONTINUE
         DO I= 1,NPE
            ELF(I)=( (1-THETA)*ELF1(I) + THETA*ELF2(I) )*DT
         ENDDO
         CALL ELFT    ( NE     , NPE   , DT   , THETA , ELSTIF , ELF   ,
     1                  DMASS , CPHIO, NOD , NELM ,  NSIZ              )
         CALL ADDSF   ( NPE,DT,THETA,ELSTIF,DMASS,NDFC)
```

```
        CALL FRONT
     1( ELSTIF    , ELF    , NE    , NOP  , NELM    , NSTF , LDEST , NK      ,
     2  MAXFR     , EQ     , LHED  , LPIV , JMOD    , QQ   , PVKOL , CPHI    ,
     3  R1        , NCOD   , BC    , NOPP , MDF     , NDNC , NSIZ  , NEM     ,
     4  NSIZ      , NEC    , LCOL  , NELL , NPE                             )
  34  CONTINUE
C.......................
      RETURN
      END
C..........................................
C     CALCULATION OF STREAM FUNCTION             .
C..........................................
      SUBROUTINE STRMFC (
     1  SRF    , GF                                                        ,
     2  GAUSS  , WT     , NSB    , ISSB   , NSSB                           ,
     3  X      , Y      , NOD    , NOP                                     ,
     4  BCS    , NCODS  , NOPPS  , MDFS   , NDNS                           ,
     5  NPE    , IR     , IF                      , NDFS , NEM             ,
     6  NES    , NNM                                                       ,
     7  NSIZ   , NSTF   , NELM   , NNOD   , MAXFR                          ,
     8  R1     , ELF    , ELSTIF                                           ,
     9  LDEST  , NK     , EQ     , LHED   , LPIV                           ,
     1  JMOD   , QQ     , PVKOL                                            ,
     2  NCOD   , BC     , NOPP   , MDF                                     )
C..............................
      IMPLICIT REAL*8 (A-H,O-Z)
C..............................
      DIMENSION SRF   ( NSIZ ) , GF ( NSIZ )
      DIMENSION GAUSS ( 7,7  ) , WT( 7,7 )
      DIMENSION X     ( NNOD ) , Y ( NNOD )
      DIMENSION NOD   ( NELM   , 9 )            , NOP ( NELM , 9)
      DIMENSION BCS   ( NSIZ ) , NCODS ( NSIZ ) , NOPPS ( NSIZ )
      DIMENSION MDFS  ( NSIZ ) , NDNS  ( NSIZ )
      DIMENSION ISSB  ( NSIZ ) , NSSB  ( NSIZ , 3 )
C...........................
      DIMENSION ELF   ( NSTF )  , ELSTIF ( NSTF,NSTF )
      DIMENSION LDEST ( NSTF  )
      DIMENSION LHED  ( MAXFR )
      DIMENSION NK    ( NSTF  )
      DIMENSION LPIV  ( MAXFR )
      DIMENSION JMOD  ( MAXFR ) , QQ    ( MAXFR )
      DIMENSION PVKOL ( MAXFR ) , R1 ( NSIZ )
      DIMENSION EQ    ( MAXFR   , MAXFR )
      DIMENSION BC    ( NSIZ  ) , NCOD ( NSIZ ) , NOPP ( NSIZ )
      DIMENSION MDF   ( NSIZ  )
C.........................
      CALL ARRZRF ( SRF, NES  )
      CALL ARRZRF ( R1 , NSIZ )
C ....
C ....
      CALL RSAVE ( BC   , BCS   , NSIZ )
      CALL RSAVI ( NCOD , NCODS , NSIZ )
      CALL RSAVI ( NOPP , NOPPS , NSIZ )
      CALL RSAVI ( MDF  , MDFS  , NSIZ )
C.....
```

```
      DO 34 NE=1,NEM
         CALL STIFFS   ( NE    , NPE   , GAUSS  , WT  , ELSTIF  , GF    ,
     1                   NOD   , X     , Y      , IR  , IF      , NELM  ,
     2                   NNOD  , NSIZ  , NSTF   , ELF, ISSB     , NSSB  , NSB )
         CALL FRONT
     1( ELSTIF    , ELF    , NE    , NOP  , NELM   , NSTF , LDEST , NK     ,
     2  MAXFR     , EQ     , LHED  , LPIV , JMOD   , QQ   , PVKOL , SRF    ,
     3  R1        , NCOD   , BC    , NOPP , MDF    , NDNS , NSIZ  , NEM    ,
     4  NSIZ      , NES    , LCOL  , NELL , NPE                            )
   34 CONTINUE
      RETURN
      END
C.....................................................
C     CALCULATION OF VELOCITY COMPONENTS USING GLOBAL .
C     VECTOR ON THE ELEMENT SIDE                      .
C.....................................................
      SUBROUTINE UVN1D ( NE , NOD , NSDOF , SI , G , NSIZ , NELM ,
     1                   VX , VY  , IF    , I                      )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD   ( NELM , 9 ) , G ( NSIZ ) , SI (3)
      DIMENSION NSDOF ( NSIZ , 3 )
      VX  = 0.0
      VY  = 0.0
      DO L=1,IF
         NN1  = NOD ( NE , NSDOF ( I,L) )
         NNX  = 2*NN1 - 1
         NNY  = 2*NN1
         VX = VX + SI(L) * G  ( NNX )
         VY = VY + SI(L) * G  ( NNY )

      ENDDO
      VX=1
      VY=1
      RETURN
      END
C......................................................
C  ELEMENT STIFFNESS MATRIX FOR STREAM FUNCTION        .
C......................................................
      SUBROUTINE STIFFS   ( NE    , NPE   , GAUSS  , WT , ELSTIF , GF    ,
     1                      NOD   , X     , Y      , IR , IF     , NELM  ,
     2                      NNOD  , NSIZ  , NSTF   , ELF, ISSB   , NSSB  , NSB)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD ( NELM,9 ) , X (NNOD) , Y(NNOD)                     ,
     1          GAUSS (7,7), WT(7,7)    , ELSTIF(18,18) , GF(NSIZ)     ,
     3          DSIK(9) , DSIE(9) , SI(9), DSIKM(9),DSIEM(9),SIM(9) ,
     4          XJ(9)        , YJ(9)      , NP(3)                       ,
     5          AJI(2,2)     , AJ(2,2)    , ELF ( NSTF )                ,
     6          ISSB ( NSIZ ) , NSSB ( NSIZ , 3 )                       ,
     7          SI1D (3)      , DSI1D(3)
C..............................................................
      CALL ARR2ZF  ( ELSTIF , NSTF )
      CALL ARRZRF  ( ELF    , NSTF )
C.........
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
```

```
            YJ(I)=Y(NOD(NE,I))
         ENDDO
C
C ..............................................
C
         DO 14 KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO 14 KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
         1  DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)



               CALL JACOB2 ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
        1                        NPE , NE )
               GUY = 0.0
               GVX = 0.0
               DO L = 1,NPE
                  DFX = GF (2*NOD(NE,L)-1)
                  DFY = GF (2*NOD(NE,L))
                  RUY = DFX * ( DSIK(L) * AJI(2,1) + DSIE(L) * AJI(2,2) )
                  RVX = DFY * ( DSIK(L) * AJI(1,1) + DSIE(L) * AJI(1,2) )
                  GUY = GUY + RUY
                  GVX = GVX + RVX
               ENDDO
               DO 16 M=1,NPE
                  DSXM= DSIK(M) * AJI(1,1) + DSIE(M) * AJI(1,2)
                  DSYM= DSIK(M) * AJI(2,1) + DSIE(M) * AJI(2,2)
                  DO 18 N=1,NPE
                      DSXN= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                      DSYN= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)
                      ELSTIF ( M,N ) = ELSTIF ( M,N ) +
        1                                 ( DSXM  * DSXN + DSYM  * DSYN )
        2                                *  DET*WT(KI,IF)*WT(KJ,IF)
 18               CONTINUE
               ELF (M) = ELF (M) + (-1)*(SI(M)*GUY-SI(M)*GVX)*
        1                                   DET*WT(KI,IF)*WT(KJ,IF)
 16            CONTINUE
 14      CONTINUE
C....
C............CALCULATION OF BOUNDARY INTEGRAL

         DO I = 1,NSB
C..................
            IF ( NE .NE. ISSB (I) ) GOTO 100

C......CALCULATION OF THE COMPONENTS OF THE UNIT VECTOR NORMAL TO ELEMENT
SIDE.

            NP(3)= NOD ( NE , NSSB ( I,3 ) )
            NP(2)= NOD ( NE , NSSB ( I,2 ) )
            NP(1)= NOD ( NE , NSSB ( I,1 ) )
            XL = X (NP(3)) - X(NP(1))
            YL = Y (NP(3)) - Y(NP(1))
```

```
          ELLGTH = DSQRT ( XL**2+YL**2)
          DNX = YL /ELLGTH
          DNY =-XL/ELLGTH
C.......................................
          DO K = 1,IF
             AKESI = GAUSS (K,IF)
             CALL LAGSH1 ( AKESI , SI1D , DSI1D )
C.............................................
             CALL UVN1D ( NE , NOD , NSSB   , SI1D , GF, NSIZ , NELM ,
     1                    VX , VY  , IF     , I                      )
C..............................................
             DO J=1,IF
C...         CALL UTNML ( J , X , Y , NP , DNX , DNY, ELLGTH, NNOD ,
C... 1                    NSSB   , NSIZ   , I                        )
                NDFQ = NSSB(I,J)
                DJACOB = ELLGTH / 2.0
                ELF( NDFQ ) =ELF( NDFQ ) + (   SI1D(J)*(-VY*DNX+VX*DNY)*
     1                          ( DJACOB   ) * WT (K,IF)    )
             ENDDO
          ENDDO
 100  ENDDO
C.................................................................
      RETURN
      END
C............................................................
C     SOLUTION OF FLOW EQUATIONS , CYLINDRICAL COORDINATES    .
C     WITH  CEF RHEOLOGICAL EQUATION                          .
C............................................................
      SUBROUTINE FLCYL   (
     1  GF     , GFI    , TFI     , SO     , GFSI   , GFSO , TO          ,
     2  GAUSS  , WT     , CPHI                                         ,
     3  X      , Y      , NOD     , NOP                                ,
     4  BCV    , NCODV  , NOPPV   , MDFV   , NDNV                      ,
     5  NPE    , IR     , IF      , DT     , THETA  , NDFV , NEM       ,
     6  NEQ    , NNM    , NTRAN   , NCARB                              ,
     7  VPROP  , IVIS                                                  ,
     8  NSIZ   , NSTF   , NELM    , NNOD   , MAXFR                     ,
     9  R1     , ELF    , ELSTIF  , DMASS  , VHS                       ,
     1  LDEST  , NK     , EQ      , LHED   , LPIV                      ,
     2  JMOD   , QQ     , PVKOL                                        ,
     3  NCOD   , BC     , NOPP    , MDF    , NSDOF  , PPVL , NBF       )
C..............................
      IMPLICIT REAL*8 (A-H,O-Z)
C..............................
      DIMENSION GF     ( NSIZ ) , GFI ( NSIZ ) , TFI ( NSIZ )
      DIMENSION SO     ( NSIZ ) , VHS ( NSIZ , 5)   , TO (NSIZ)
      DIMENSION GFSI   ( NSIZ ) , GFSO ( NSIZ )
      DIMENSION CPHI   ( NSIZ )
      DIMENSION GAUSS  ( 7,7  ) , WT( 7,7 )
      DIMENSION VPROP  ( 30   )
      DIMENSION X      ( NNOD ) , Y ( NNOD )
      DIMENSION NOD    ( NELM   , 9 )            , NOP ( NELM , 9)
      DIMENSION BCV    ( NSIZ ) , NCODV ( NSIZ ) , NOPPV ( NSIZ )
      DIMENSION MDFV   ( NSIZ ) , NDNV   ( NSIZ )
      DIMENSION NSDOF  ( NSIZ , 4 ) , PPVL ( NSIZ )
```

```
C.............................
      DIMENSION ELF    ( NSTF )   , ELSTIF ( NSTF,NSTF )
      DIMENSION ELSTID ( 18    , 18    )
      DIMENSION DMASS  ( NSTF      , NSTF )
      DIMENSION LDEST  ( NSTF   )
      DIMENSION LHED   ( MAXFR )
      DIMENSION NK     ( NSTF   )
      DIMENSION LPIV   ( MAXFR )
      DIMENSION JMOD   ( MAXFR ) , QQ     ( MAXFR )
      DIMENSION PVKOL  ( MAXFR ) , R1  ( NSIZ )
      DIMENSION EQ     ( MAXFR    , MAXFR )
      DIMENSION BC     ( NSIZ   ) , NCOD  ( NSIZ ) , NOPP  ( NSIZ )
      DIMENSION MDF    ( NSIZ   )
C..............................
      CALL ARRZRF ( GF    , NEQ    )
      CALL ARRZRF ( R1    , NSIZ )
C...
C...
      CALL RSAVI   ( NOPP    , NOPPV    , NSIZ )
      CALL RSAVI   ( MDF     , MDFV     , NSIZ )
      CALL ARRZRI ( NCOD    , NSIZ )
      CALL ARRZRF ( BC      , NSIZ )
C...
      DO 20 NE=1,NEM
         CALL   ARRZRF ( ELF , NSTF )
         CALL    STICYL( NE     , NPE , GAUSS , WT    , ELSTIF           ,
     1                   VPROP , GFI , TFI    , VHS   , CPHI             ,
     2                   NOD    , X   , Y     , IR , IF    , NELM   , NNOD,
     3                   NSIZ   , NSTF, IVIS  , GFSI , NCARB            )
C....
         IF ( NTRAN .EQ.2.OR.NTRAN.EQ.3 ) THEN
         IF ( THETA.EQ.1.0 ) THEN
         CALL ARR2ZF ( ELSTID , NSTF )
         ELSE
         CALL    STICYL( NE     , NPE , GAUSS , WT    , ELSTID           ,
     1                   VPROP , SO   , TO    , VHS   , CPHI             ,
     2                   NOD    , X   , Y     , IR , IF    , NELM   , NNOD,
     3                   NSIZ   , NSTF, IVIS  , GFSO   , NCARB           )
C...
         ENDIF
            CALL MASCYL ( NE  , NPE , GAUSS , WT , DMASS, NOD  ,
     1                     X   , Y    , IR     , IF , NELM , NNOD , NSTF  ,
     2                     GFSI , VPROP , NSIZ                              )
            CALL ELFC    ( NE     , NPE , DT      , THETA , ELSTID , ELF   ,
     1                     DMASS , SO   , NOD     , NELM   , NSIZ           )
            CALL ADDSF   ( NPE , DT , THETA , ELSTIF , DMASS , NDFV )
         ENDIF
         CALL   PUBCLV ( NOD    , NCODV , BCV    , ELSTIF , ELF          ,
     1                   NSDOF, NE      , NSIZ , NSTF     , NELM , NNOD ,
     2                   X      , Y     , PPVL , NPE      , NBF   , VPROP,
     3                   GFI    , TFI   , IVIS , VHS      , CPHI , NCARB,
     4                   GFSI , IR      , IF    , GAUSS    , WT            )
C...
         CALL FRONT
     1( ELSTIF    , ELF    , NE    , NOP   , NELM    , NSTF , LDEST , NK     ,
     2  MAXFR     , EQ     , LHED , LPIV , JMOD  , QQ   , PVKOL , GF    ,
     3  R1        , NCOD   , BC   , NOPP , MDF   , NDNV , NSIZ  , NEM   ,
     4  NSIZ      , NEQ    , LCOL , NELL , NPE                          )
C...
 20   CONTINUE
C....................
      RETURN
C....................
      END
C.........................................................
C     CALCULATION OF ELEMENT STIFFNESS MATRIX  FOR FLOW EQ.    .
C.........................................................
C
      SUBROUTINE STICYL( NE    , NPE , GAUSS ,WT   , ELSTIF          ,
     1                   VPROP , GFI , T     ,VHS  , CPHI            ,
     2                   NOD   , X   , Y     ,IR ,IF    , NELM   , NNOD,
     3                   NSIZ  , NSTF, IVIS  , GFSI   , NCARB        )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  NOD   (NELM,9) , X (NNOD) , Y (NNOD)
      DIMENSION  VPROP ( 30    ) , VHS ( NSIZ , 5 ) , CPHI ( NSIZ )
      DIMENSION  GFI (NSIZ)    , T(NSIZ)    , GFSI ( NSIZ )
      DIMENSION  AK11 (9,9) , AK12 (9,9) , AK21 (9,9) , AK22 (9,9)
      DIMENSION  S11  (9,9) , S12  (9,9) , S21  (9,9) , S22 (9,9)
      DIMENSION  C11  (9,9) , C12  (9,9) , C21  (9,9) , C22 (9,9)
      DIMENSION  DSIK (9), DSIE (9), SI (9),DSIKM(9),DSIEM(9),SIM(9)
      DIMENSION  XJ   (9)    , YJ   (9)    , AJ   (2,2) , AJI  (2,2)
      DIMENSION  ELSTIF (18,18) , GAUSS (7,7) , WT  (7,7)
C.............
      CALL ARR2ZF ( AK11 , 9   )
      CALL ARR2ZF ( AK12 , 9   )
      CALL ARR2ZF ( AK21 , 9   )
      CALL ARR2ZF ( AK22 , 9   )
      CALL ARR2ZF ( S11  , 9   )
      CALL ARR2ZF ( S12  , 9   )
      CALL ARR2ZF ( S21  , 9   )
      CALL ARR2ZF ( S22  , 9   )
      CALL ARR2ZF ( C11  , 9   )
      CALL ARR2ZF ( C12  , 9   )
      CALL ARR2ZF ( C21  , 9   )
      CALL ARR2ZF ( C22  , 9   )
      CALL ARR2ZF ( ELSTIF , NSTF )
C.............
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
C     FULL INTEGRATION AND CONVECTION TERMS
C
      DO 24 KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO 24 KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1          DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
```

```
            CALL JACCYL ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                   NPE , NE )
            CALL VISCEF ( AMU    , GAMAD , VPROP, NE    , CPHI ,  NPE ,
     1                   AKESI , ETA   , GFI  , T     , NOD  ,  Y   ,
     2                   X     , NELM  , NNOD , NSIZ  , IVIS , PSIC ,
     3                   GUX   , GUY   , GVX  , GVY   , UN   , VN   ,
     4                   XC    , YC    , ELVIS , NCARB              )
            CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
            AMU = FVAL*AMU +(1-FVAL)*VPROP(8)
            DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
            PSIC= FVAL*PSIC
            COEF= DET*WT(KI,IF)*WT(KJ,IF)
            DO 26 M=1,NPE
               DSXM= DSIKM(M) * AJI(1,1) + DSIEM(M) * AJI(1,2)
               DSYM= DSIKM(M) * AJI(2,1) + DSIEM(M) * AJI(2,2)
               CALL RTDER ( DSRM, DSTM , DSXM , DSYM , XC , YC )
               DO 26 N=1,NPE
                  DSXN= DSIK(N) * AJI(1,1) + DSIE(N) * AJI(1,2)
                  DSYN= DSIK(N) * AJI(2,1) + DSIE(N) * AJI(2,2)
                  CALL RTDER ( DSRN, DSTN , DSXN , DSYN , XC , YC )
                  APP = GVX-(VN/XC)+(1./XC)*GUY
                  S11(M,N)=  S11(M,N)              +
     1                     ( DEN*UN*SI(M)*DSRN     +
     2                       DEN*(VN/XC)*SI(M)*DSTN   +
     3                       2*ELVIS*DSRM*DSRN        +
     4                       (AMU/XC**2)*DSTM*DSTN    +
     5                       (2*ELVIS/(XC**2))*SI(M)*SI(N) +
     6                       PSIC*APP*(SI(M)/XC**2)*DSTN )*COEF
                  S12(M,N)=  S12(M,N)                +
     1                     ( -DEN*(VN/XC)*SI(M)*SI(N)  +
     2                       (AMU/XC)*DSTM*DSRN       -
     3                       (AMU/XC**2)*DSTM*SI(N)    +
     4                       (2*ELVIS/(XC**2))*SI(M)*DSTN +
     5                       PSIC*APP*
     6                       ((SIM(M)/XC)*DSRN-SI(M)*(SI(N)/XC**2))
     7                       ) * COEF
                  S21(M,N)=  S21(M,N)                 +
     1                     ( DEN*(VN/XC)*SI(M)*SI(N)     +
     2                       (AMU/XC)*(DSRM*DSTN-(SI(M)/XC)*DSTN)+
     3                       ( (2*ELVIS/XC**2)*DSTM*SI(N) )       +
     4                       PSIC*APP*(1./XC**2)*DSTM*DSTN
     5                     ) * COEF
                  S22(M,N)=  S22(M,N)                 +
     1                   ( DEN*UN*SIM(M)*DSRN+DEN*VN*SI(M)*DSTN/XC+
     2                       AMU*(DSRM*DSRN-SI(M)*DSRN/XC-
     3                       DSRM*SI(N)/XC+SI(M)*SI(N)/XC**2)      +
     4                       (2*ELVIS/XC**2)*DSTM*DSTN             +
     5                       (PSIC/XC)*APP*
     6                       (DSTM*DSRN-DSTM*SI(N)/XC)
     7                     ) * COEF
 26            CONTINUE
 24         CONTINUE
C
C     REDUCED INTEGRATION
C
```

```
C..........
      DO 56 KI=1,IR
         AKESI=GAUSS(KI,IR)
         DO 56 KJ=1,IR
            ETA=GAUSS(KJ,IR)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1          DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
               CALL JACCYL ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                       NPE , NE )
               CALL VISCEF ( AMU   , GAMAD , VPROP, NE   , CPHI ,  NPE ,
     1                       AKESI , ETA   , GFI  , T    , NOD  ,  Y   ,
     2                       X     , NELM  , NNOD , NSIZ , IVIS , PSIC ,
     3                       GUX   , GUY   , GVX  , GVY  , UN   , VN   ,
     4                       XC    , YC    , ELVIS  , NCARB            )
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
               AMU = FVAL*AMU+(1-FVAL)*VPROP(8)
               COEF= VPROP(15)*AMU*DET*WT(KI,IR)*WT(KJ,IR)
               DO 30 M=1,NPE
                  DSXM= DSIK(M)  * AJI(1,1) + DSIE(M)  * AJI(1,2)
                  DSYM= DSIK(M)  * AJI(2,1) + DSIE(M)  * AJI(2,2)
                  CALL RTDER ( DSRM, DSTM , DSXM , DSYM , XC , YC )
                  DO 30 N=1,NPE
                     DSXN= DSIK(N)  * AJI(1,1)  + DSIE(N)  * AJI(1,2)
                     DSYN= DSIK(N)  * AJI(2,1)  + DSIE(N)  * AJI(2,2)
                     CALL RTDER ( DSRN, DSTN , DSXN , DSYN , XC , YC )
                     C11(M,N)=  C11(M,N) +
     1                         ( DSRM+SI(M)/XC )*( DSRN+SI(N)/XC )
     2                         * COEF
                     C12(M,N)=  C12(M,N) +
     1                          ( DSRM/XC+SI(M)/XC**2 ) * DSTN * COEF
                     C21(M,N)=  C21(M,N) +
     1                          ( DSTM*DSRN/XC+DSTM*SI(N)/XC**2) * COEF
                     C22(M,N)=  C22(M,N) + ( DSTM*DSTN/XC**2) * COEF
 30            CONTINUE
 56   CONTINUE
C
C.......................
        DO 32 I=1,NPE
           DO 32 J=1,NPE
              AK11(I,J)= S11(I,J) + C11(I,J)
              AK12(I,J)= S12(I,J) + C12(I,J)
              AK21(I,J)= S21(I,J) + C21(I,J)
 32           AK22(I,J)= S22(I,J) + C22(I,J)
C
C     REORDERING  THE STIFFNESS MATRIX
C
      DO I=1,NPE
         M=2*I-1
         DO J=1,NPE
            N=2*J-1
            ELSTIF (M   , N   ) = AK11 (I,J)
            ELSTIF (M   , N+1 ) = AK12 (I,J)
            ELSTIF (M+1, N    ) = AK21 (I,J)
            ELSTIF (M+1, N+1 ) = AK22 (I,J)
        ENDDO
      ENDDO
C.....................................................
      RETURN
      END
C..................................................
C    CALCULATION OF VISCOSITY ( CEF MODEL  )          .
C..................................................
      SUBROUTINE VISCEF ( AMU    , GAMAD , VPROP, NE    , CPHI ,  NPE ,
     1                    AKESI  , ETA   , G    , T     , NOD  ,  Y   ,
     2                    X      , NELM  , NNOD , NSIZ  , IVIS , PSIC ,
     3                    GUX    , GUY   , GVX  , GVY   , UN   , VN   ,
     4                    XC     , YC    , ELVIS  , NCARB             )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD    ( NELM,9)  , X(NNOD)  , Y(NNOD)
      DIMENSION G      ( NSIZ  )  , T(NSIZ)  , CPHI ( NSIZ )
      DIMENSION VPROP ( 30     )
      DIMENSION DSIK(9) , DSIE(9) , SI(9), DSIKM(9),DSIEM(9),SIM(9) ,
     1          XJ(9)   , YJ(9)            ,
     2          AJ(2,2) , AJI(2,2)
C......................................
      CALL SHAPE ( AKESI  ,ETA , DSIK , DSIE , SI , NPE ,
      1      DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
      CALL UVN   ( UN    , VN , SI ,SIM , NPE   , NE , G   , NOD ,
     1             NELM , NSIZ             )
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
      CALL JACCYL ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1              NPE , NE )
      CALL RTFIND ( AKESI , ETA , NPE , XJ , YJ, XC ,YC )
      GUX= 0.0
      GUY= 0.0
      GVX= 0.0
      GVY= 0.0
      GT = 0.0
      DO I=1,NPE
         INN = NOD(NE,I)
         DFT = T    ( INN        )
         DFX = G    ( 2*INN - 1 )
         DFY = G    ( 2*INN      )
         DSX = DSIK(I) * AJI(1,1) + DSIE(I) * AJI(1,2)
         DSY = DSIK(I) * AJI(2,1) + DSIE(I) * AJI(2,2)
         CALL RTDER ( DSR , DST   , DSX   , DSY   , XC , YC )
         RUX = DFX * DSR
         RUY = DFX * DST
         RVX = DFY * DSR
         RVY = DFY * DST
         GUX = GUX + RUX
         GUY = GUY + RUY
         GVX = GVX + RVX
         GVY = GVY + RVY
         GT  = GT  + DFT*SI(I)
      ENDDO
C
```

```
C.... AII IS THE SECOND INVARIANT OF R.D.T BASED ON THE DIRECT CALCULATION
C     OF VELOCITY GRADIENT COMPONENTS
C
      DRR = 2 * GUX
      DRT = GVX - (VN/XC) + (1./XC)*GUY
      DTT = 2*( (1./XC)*GVY + (UN/XC) )
      AII=DRR**2 + DTT**2 + 2.0*DRT**2
      GAMAD=DSQRT(0.5*AII)
      CALL VISEQU ( AMU , GAMAD , GT , VPROP , IVIS )
      ELVIS= 6*AMU
      PS11 = AMU**VPROP(13)
      PS22 = (VPROP(13)-2)/2.0
      PSIC = VPROP(12)*PS11
      IF (AII.NE.0) PSIC = PSIC*((0.5*AII)**PS22)
C.............................................................
C     MODIFICATION OF VISCOSITY TO INCLUDE THE EFFECT OF THE .
C     EFFECTIVE FILLER VOLUME FRACTION                       .
C.............................................................
      IF ( NCARB.EQ.2 ) THEN
         CPP =0.0
         DO I=1,NPE
            CPP=CPP+ SI(I)*CPHI ( NOD(NE,I) )
         ENDDO
         CALL BOUN01 ( CPP )
         RELVIS = VPROP(21)+VPROP(22)*CPP
         AMU = AMU * RELVIS
      ENDIF
      RETURN
      END
C...........................................................
C     CALCULATION OF JACOBIAN   ( CYLINDRICAL COORDINATE )   .
C...........................................................
      SUBROUTINE JACCYL ( AJ  , AJI  , DET , X , Y ,
     1                   DSIK , DSIE , N   , NE )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AJ (2,2) , AJI (2,2) ,
     1          X(N)     , Y(N)      ,
     2          DSIK(N)  , DSIE(N)   ,
     3          XJJ (9)  , YJJ(9)
      DO I=1,2
         DO J=1,2
            AJ  (I,J)  = 0.0
            AJI (I,J)  = 0.0
         ENDDO
      ENDDO
C
C... CHANGE OF COORDINATES FROM CYLINDRICAL TO CARTESIAN
C
      DO I=1,N
         CALL CYL2CAR ( XJJ(I) , YJJ(I) , X(I) ,Y(I) )
      ENDDO
C.....................................................
      DO I=1,N
         AJ(1,1) = AJ(1,1) + XJJ(I) * DSIK(I)
         AJ(1,2) = AJ(1,2) + YJJ(I) * DSIK(I)
         AJ(2,1) = AJ(2,1) + XJJ(I) * DSIE(I)
         AJ(2,2) = AJ(2,2) + YJJ(I) * DSIE(I)
      ENDDO
      DET=AJ(1,1)*AJ(2,2)-AJ(1,2)*AJ(2,1)
      IF (DET.LE.0.0) THEN
         WRITE (2,110) NE,DET
 110     FORMAT  (1X,'ERROR : ZERRO OR NEGATIVE JACOBIAN=',I6,G20.5)
         STOP
      ENDIF
      AJI(1,1) =  AJ(2,2) / DET
      AJI(1,2) = -AJ(1,2) / DET
      AJI(2,1) = -AJ(2,1) / DET
      AJI(2,2) =  AJ(1,1) / DET
      RETURN
      END
C....................................
C     CALCULATION OF FIRST DERIVATIVES  .
C....................................
      SUBROUTINE RTDER ( DSR , DST , DSX , DSY , R , T )
      IMPLICIT REAL*8 (A-H,O-Z)
      DSR = DSX * DCOS(T) + DSY * DSIN (T)
      DST = DSX * (-R)*(DSIN(T)) + DSY * R*DCOS(T)
      RETURN
      END
C.............................................................
C     FIND THE VALUE OF RADIUS AND THETA AT INTEGRATION POINTS .
C.............................................................
      SUBROUTINE RTFIND ( AKESI , ETA , NPE , XJ , YJ, R  ,T  )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XJ(9)   , YJ(9)   , XJJ(9) , YJJ(9)
      DIMENSION DSIK(9) , DSIE(9) , SI(9),DSIKM(9),DSIEM(9),SIM(9)
      DO I=1,NPE
         CALL CYL2CAR ( XJJ(I) , YJJ(I) , XJ(I) ,YJ(I) )
      ENDDO
      X=0.0
      Y=0.0
      CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1     DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X   , Y ,NELM, NNOD)
      DO I=1,NPE
         X=X+XJJ(I)*SI(I)
         Y=Y+YJJ(I)*SI(I)
      ENDDO
      CALL CAR2CYL ( X , Y , R , T )
      RETURN
      END
C..........................................................
C     COORDINATE TRANSFORMATION FROM CYLINDRICAL TO CARTESIAN .
C     SYSTEM                                                  .
C..........................................................
      SUBROUTINE CYL2CAR ( X , Y , R , T )
      IMPLICIT REAL*8 (A-H,O-Z)
      X = R *  DCOS (T)
      Y = R *  DSIN (T)
      RETURN
      END
```

```
C...............................................................
C     COORDINATE TRANSFORMATION FROM CARTESIAN TO CYLINDRICAL .
C     SYSTEM                                                  .
C...............................................................
      SUBROUTINE CAR2CYL ( X , Y , R , T )
      IMPLICIT REAL*8 (A-H,O-Z)
      PI= 3.141592654
      R=DSQRT ( X**2+Y**2)
      IF (DABS(X).EQ.0.0.AND.Y.GT.0.0 ) THEN
          T=PI/2.0
      ELSEIF (DABS(X).EQ.0.0.AND.Y.LT.0.0) THEN
          T=3*PI/2.0
      ELSEIF (R.EQ.0.0) THEN
          T=0.0
      ELSE
          T=DATAN(Y/X)
          IF ( X.LT.0.0 ) T=PI+T
          IF ( X.GT.0.0.AND.Y.LT.0.0 ) T=2*PI+T
      ENDIF
      RETURN
      END
C.............................................................
C     MASS MATRIX CALCULATION  (CYLINDRICAL COORDINATE )      .
C.............................................................
      SUBROUTINE MASCYL ( NE , NPE , GAUSS , WT , DMASS, NOD   ,
     1                    X   , Y    , IR     , IF , NELM , NNOD , NSTF ,
     2                    GFSI , VPROP , NSIZ                          )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION GAUSS(7,7)   , WT(7,7) , DMASS  (18,18) , DM(9,9) ,
     1          SI(9), DSIE(9) , DSIK(9), DSIKM(9),DSIEM(9),SIM(9) ,
     2          YJ(9) ,  XJ(9)     , AJ(2,2) , AJI(2,2)
      DIMENSION NOD(NELM,9) , X(NNOD) , Y(NNOD)
      DIMENSION GFSI ( NSIZ ) , VPROP ( 30 )
      NDE= 2*NPE
      DO I= 1,NPE
         XJ(I)= X(NOD(NE,I))
         YJ(I)= Y(NOD(NE,I))
      ENDDO
C..........
      CALL ARR2ZF ( DMASS , NSTF )
      CALL ARR2ZF ( DM    , 9    )
C..........
      DO 24 KI=1,IF
            AKESI=GAUSS(KI,IF)
            DO 24 KJ=1,IF
               ETA=GAUSS(KJ,IF)
               CALL SHAPE  ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
     1   DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL JACCYL ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                    NPE , NE )
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFSI, NOD, NELM, NSIZ )
               DEN = FVAL*VPROP(14)+(1-FVAL)*VPROP(9)
               DO 26 M=1,NPE
               DO 26 N=1,NPE
 26            DM(M,N)= DM(M,N)+DEN *SI(M)*SI(N)*DET*WT(KI,IF)*WT(KJ,IF)
 24         CONTINUE
C.... REORDERING  THE MASS MATRIX                .
      DO 44 I=1,NPE
         M=2*I-1
         DO 44 J=1,NPE
            N=2*J-1
            DMASS(M,N)=DM(I,J)
            DMASS(M,N+1)=0.0
            DMASS(M+1,N)=0.0
44          DMASS(M+1,N+1)=DM(I,J)
C.........................................
      CALL LUMP ( DMASS , NSTF , NDE )
C.........................................
      RETURN
      END
C.......................................................
C     CALCULATION OF PRESSURE  (CEF IN CYLINDRICAL  )  .
C.......................................................
      SUBROUTINE PRSCYL ( NEM    , GAUSS , NPE   , GF     , NOD            ,
     1                    X      , Y     , IR    , IF     , NELM , NNOD    ,
     2                    NSIZ   , VPROP , T     , CPHI   , NCARB, IVIS    ,
     3                    PRHS   , WT    , GFS   , PMG    , NNM            )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD (NELM,9)  , X(NNOD) , Y(NNOD)  ,
     1          GF   (NSIZ)    , T ( NSIZ )          ,
     2          GAUSS(7,7)     , WT (7,7)            ,
     3          DSIK(9) , DSIE(9) , SI(9), DSIKM(9),DSIEM(9),SIM(9)     ,
     4          XJ(9)          , YJ(9)               ,
     5          AJ(2,2)        , AJI(2,2)            ,
     8          PRHS  ( NSIZ ) , PMG ( NSIZ )        ,
     9          VPROP ( 30   )                       ,
     1          GFS   ( NSIZ ) , CPHI ( NSIZ )
C...............................
      DO 40  NE=1,NEM
         DO I=1,NPE
            XJ(I)=X(NOD(NE,I))
            YJ(I)=Y(NOD(NE,I))
         ENDDO
         DO 70  II=1,IR
            DO 70  JJ=1,IR
               AKESI=GAUSS(II,IR)
               ETA=GAUSS(JJ,IR)
               CALL SHAPE   ( AKESI , ETA , DSIK , DSIE , SI , NPE ,
      1       DSIKM , DSIEM , SIM,ELLGTH,NE , NOD , X    , Y ,NELM, NNOD)
               CALL JACCYL ( AJ , AJI , DET , XJ , YJ , DSIK , DSIE ,
     1                          NPE , NE )
               CALL VISCEF ( AMU    , GAMAD , VPROP, NE    , CPHI ,  NPE ,
     1                       AKESI  , ETA   , GF    , T     , NOD  ,  Y   ,
     2                       X      , NELM  , NNOD  , NSIZ  , IVIS , PSIC ,
     3                       GUX    , GUY   , GVX   , GVY   , UN   , VN   ,
     4                       XC     , YC    , ELVIS , NCARB                )
               CALL FPSL ( FVAL ,SI ,NPE  ,NE, GFS , NOD, NELM, NSIZ )
               AMU  = FVAL*AMU+(1-FVAL)*VPROP(8)
               PRESSE=-VPROP(15)*AMU*( GUX + UN/XC + (1.0/XC)*GVY )
C.............................................................
```

```
              DO I= 1,NPE
                 PRHS ( NOD (NE,I) ) = PRHS ( NOD (NE,I) ) + PRESSE*
     1           SI(I)*DET*WT(II,IR)*WT(JJ,IR)
              ENDDO
C...............................................................
70        CONTINUE
40      CONTINUE
C...............................................................
        DO I=1,NNM
          PRHS (I) =PRHS (I) / PMG (I)
        ENDDO
        RETURN
        END
C...................................................................
C       TRANSFORMATION OF A VECTOR COMPONENTS FROM CARTESIAN          .
C       TO CYLINDRICAL                                                .
C...................................................................
        SUBROUTINE VCA2CL ( G1 , G2 , THH )
        IMPLICIT REAL*8 (A-H,O-Z)
        GX=G1
        GY=G2
        G1 = GX *   DCOS(THH)  + GY * DSIN(THH)
        G2 = GX * (-DSIN(THH)) + GY * DCOS(THH)
        RETURN
        END
C.........................................................
C      TRANSFORMATION OF A VECTOR COMPONENTS FROM CYLINDRICAL .
C      TO CARTESIAN                                           .
C.........................................................
        SUBROUTINE VCL2CA ( G1 , G2 , THH )
        IMPLICIT REAL*8 (A-H,O-Z)
        GX=G1
        GY=G2
        G1 = GX *   DCOS(THH)  + GY * (-DSIN(THH))
        G2 = GX *   DSIN(THH)  + GY *   DCOS(THH)
        RETURN
        END
C....................................................................
C      PUT THE FIRST AND THE THIRD DOF ON STIFFNESS EQUATIONS  (FLOW)  .
C      IN CYLINDRICAL COORDINATE                                       .
C....................................................................
        SUBROUTINE PUBCLV ( NOD   , NCODZ , BCZ  , ELSTIF , ELF        ,
     1                    NSDOF, NE    , NSIZ , NSTF   , NELM , NNOD ,
     2                    X     , Y     , PPVL , NPE    , NBF  , VPROP,
     3                    GFI   , T     , IVIS , VHS    , CPHI , NCARB,
     4                    GFSI  , IR    , IF   , GAUSS  , WT          )
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION NOD ( NELM,9 ) , X( NNOD ) , Y( NNOD )
        DIMENSION VHS ( NSIZ , 5)  , CPHI ( NSIZ )
        DIMENSION BCZ (NSIZ)      , NCODZ (NSIZ)
        DIMENSION GFSI ( NSIZ )
        DIMENSION VPROP(30)       , GFI (NSIZ ) , T ( NSIZ )
        DIMENSION ELF ( NSTF ) , ELSTIF ( NSTF , NSTF )
        DIMENSION NSDOF ( NSIZ , 4 ), PPVL ( NSIZ )
        DIMENSION DSIK(9) , DSIE(9) , SI(9)
```

```
      DIMENSION XJ(9)    , YJ(9)
      DIMENSION AJ(2,2) , AJI(2,2)
      DIMENSION NP(3)
      DIMENSION GAUSS(7,7) , WT(7,7)
C..............
C.....GVAL = 1.0D+300...........
C..............
      DO I=1,NPE
         XJ(I)=X(NOD(NE,I))
         YJ(I)=Y(NOD(NE,I))
      ENDDO
C
C.... MODIFICATION OF STIFFNESS MATRIX AND LOAD VECTOR FOR 1ST DOF
C
      DO I= 1,NPE
         KBR=NOD(NE,I)
         DO J=1,2
            MBR= 2*KBR+J-2
            LBR= 2*I+J-2
            IF ( NCODZ(MBR).EQ. 1 ) THEN
               ELSTIF (LBR,LBR) =   1.0
               ELF    (LBR    ) =   BCZ(MBR)
               DO K=1,NSTF
               IF  ( LBR.NE.K )  ELSTIF ( LBR,K ) = 0.0
               ENDDO
            ENDIF
         ENDDO
      ENDDO
      RETURN
      END
C...........................................................
C     FRONTAL SOLVER                                       .
C...........................................................
      SUBROUTINE FRONT
     1    (AA   ,RR   ,IEL  ,NOP ,MAXEL ,MAXST, LDEST, NK  ,MAXFR, EQ  ,
     2     LHED ,LPIV ,JMOD ,QQ  ,PVKOL ,DIS  , R1   , NCOD,BC    , NOPP,
     3     MDF  ,NDN  ,MAXDF,NEL ,MAXTE ,NTOV ,LCOL  , NELL,NPE         )
C
C     FRONTAL ELIMINATION ROUTINE USING DIAGONAL PIVOTING
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION AA    (MAXST,MAXST) ,RR    (MAXST)
      DIMENSION NOP   (MAXEL,9     )
      DIMENSION LDEST(MAXST)        ,NK    (MAXST)
      DIMENSION EQ    (MAXFR,MAXFR) ,LHED  (MAXFR)
      DIMENSION LPIV  (MAXFR)
      DIMENSION JMOD  (MAXFR)       ,QQ    (MAXFR)       ,PVKOL(MAXFR)
      DIMENSION DIS   (MAXTE)       ,R1    (MAXDF)       ,NCOD  (MAXDF)
      DIMENSION BC    (MAXDF)       ,NOPP  (MAXDF)       ,MDF   (MAXDF)
      DIMENSION NDN   (MAXDF)
C
      NLP=6
      ND1=14
C
C     ........
```

```
C
      NMAX=MAXFR
      NCRIT=50
      NLARG=MAXFR-10
      IF(IEL.EQ.1) NELL = 0
C   *************************
      IF(IEL.GT.1) GO TO 18
      LCOL = 0
      DO 16 I = 1,NMAX
      DO 16 J = 1,NMAX
      EQ(J,I) = 0.
   16 CONTINUE
   18 NELL = NELL+1
      N = NELL
      JDN = NDN(NELL)
      KC = 0
      DO 22 J = 1,NPE
      NN = NOP(N,J)
      M = IABS(NN)
      K = NOPP(M)
      IDF = MDF(M)
C *********************
      NR = ( M - 1 ) * IDF
C *********************
C     R1(M) = RR(J)+R1(M)
C *********************
      DO 22 L = 1,IDF
C *********************
      NR=NR+1
      NL=( J-1 )*IDF+L
      R1(NR)=R1(NR)+RR (NL)
C**********************
      KC = KC+1
      II = K+L-1
      IF(NN.LT.0)II = -II
      NK(KC) = II
   22 CONTINUE
C
C     SET UP HEADING VECTORS
C
      DO 36 LK = 1,KC
      NODE = NK(LK)
      IF(LCOL.EQ.0)GO TO 28
      DO 24 L = 1,LCOL
      LL = L
      IF(IABS(NODE).EQ.IABS(LHED(L)))GO TO 32
   24 CONTINUE
   28 LCOL = LCOL+1
      LDEST(LK) = LCOL
      LHED(LCOL) = NODE
      GO TO 36
   32 LDEST(LK) = LL
      LHED(LL) = NODE
   36 CONTINUE
      IF(LCOL.LE.NMAX)GO TO 54
      NERROR = 2
      WRITE(NLP,417)NERROR
      STOP
   54 CONTINUE
      DO 56 L = 1,KC
      LL = LDEST(L)
      DO 56 K = 1,KC
      KK = LDEST(K)
      EQ(KK,LL) = EQ(KK,LL)+AA(K,L)
   56 CONTINUE
      IF(LCOL.LT.NCRIT.AND.NELL.LT.NEL) RETURN
C
C     FIND OUT WHICH MATRIX ELEMENTS ARE FULLY ASSEMBELED
C
 60   LC = 0
      IR = 0
      DO 64 L = 1,LCOL
      KT = LHED(L)
      IF(KT.GE.0)GO TO 64
      LC = LC+1
      LPIV(LC) = L
      KRO = IABS(KT)
      IF(NCOD(KRO).NE.1)GO TO 64
      IR = IR+1
      JMOD(IR) = L
      NCOD(KRO) = 2
      R1(KRO) = BC(KRO)
   64 CONTINUE
C
C     MODIFY EQUATIONS WITH APPLIED BOUNDARY CONDITIONS
C
      IF(IR.EQ.0)GO TO 71
      DO 70 IRR = 1,IR
      K = JMOD(IRR)
      KH = IABS(LHED(K))
      DO 69 L = 1,LCOL
      EQ(K,L) = 0.
      LH = IABS(LHED(L))
      IF(LH.EQ.KH)EQ(K,L) = 1.
   69 CONTINUE
   70 CONTINUE
   71 CONTINUE
      IF(LC.GT.0)GO TO 72
      NCRIT = NCRIT+10
C     WRITE(NLP,484)NCRIT
      IF(NCRIT.LE.NLARG) RETURN
      NERROR = 3
      WRITE(NLP,418)NERROR
      STOP
   72 CONTINUE
C
C     SEARCH FOR ABSOLUTE PIVOT
C
      PIVOT = 0.
      DO 76 L = 1,LC
      LPIVC = LPIV(L)
      KPIVR = LPIVC
      PIVA = EQ(KPIVR,LPIVC)
      IF(ABS(PIVA).LT.ABS(PIVOT))GO TO 74
      PIVOT = PIVA
      LPIVCO = LPIVC
      KPIVRO = KPIVR
   74 CONTINUE
   76 CONTINUE
      IF(PIVOT.EQ.0.0)  RETURN
C
C     NORMALISE PIVOTAL ROW
C
      LCO = IABS(LHED(LPIVCO))
      KRO = LCO
C     IF(NIT.EQ.0.OR.NPRA.EQ.0)GO TO 78
C     WRITE(NLP,452)KRO,LCO,PIVOT
C  78 CONTINUE
      IF(ABS(PIVOT).LT.0.1D-08) WRITE(NLP,476)
      DO 80 L = 1,LCOL
      QQ(L) = EQ(KPIVRO,L)/PIVOT
   80 CONTINUE
      RHS = R1(KRO)/PIVOT
      R1(KRO) = RHS
      PVKOL(KPIVRO) = PIVOT
C
C     ELIMINATE THEN DELETE PIVOTAL ROW AND COLUMN
C
      IF(KPIVRO.EQ.1)GO TO 104
      KPIVR = KPIVRO-1
      DO 100 K = 1,KPIVR
      KRW = IABS(LHED(K))
      FAC = EQ(K,LPIVCO)
      PVKOL(K) = FAC
      IF(LPIVCO.EQ.1.OR.FAC.EQ.0.)GO TO 88
      LPIVC = LPIVCO-1
      DO 84 L = 1,LPIVC
      EQ(K,L) = EQ(K,L)-FAC*QQ(L)
   84 CONTINUE
   88 IF(LPIVCO.EQ.LCOL)GO TO 96
      LPIVC = LPIVCO+1
      DO 92 L = LPIVC,LCOL
      EQ(K,L-1) = EQ(K,L)-FAC*QQ(L)
   92 CONTINUE
   96 R1(KRW) = R1(KRW)-FAC*RHS
  100 CONTINUE
  104 IF(KPIVRO.EQ.LCOL)GO TO 128
      KPIVR = KPIVRO+1
      DO 124 K = KPIVR,LCOL
      KRW = IABS(LHED(K))
      FAC = EQ(K,LPIVCO)
      PVKOL(K) = FAC
      IF(LPIVCO.EQ.1)GO TO 112
      LPIVC = LPIVCO-1
      DO 108 L = 1,LPIVC
      EQ(K-1,L) = EQ(K,L)-FAC*QQ(L)
  108 CONTINUE
  112 IF(LPIVCO.EQ.LCOL)GO TO 120
      LPIVC = LPIVCO+1
      DO 116 L = LPIVC,LCOL
      EQ(K-1,L-1) = EQ(K,L)-FAC*QQ(L)
  116 CONTINUE
  120 R1(KRW) = R1(KRW)-FAC*RHS
  124 CONTINUE
  128 CONTINUE
C
C     WRITE PIVOTAL EQUATION ON DISC
C
      WRITE(ND1) KRO,LCOL,LPIVCO,(LHED(L),QQ(L),L = 1,LCOL)
      DO 130 L = 1,LCOL
      EQ(L,LCOL) = 0.
      EQ(LCOL,L) = 0.
  130 CONTINUE
C
C     REARRANGE HEADING VECTORS
C
      LCOL = LCOL-1
      IF(LPIVCO.EQ.LCOL+1)GO TO 136
      DO 132 L = LPIVCO,LCOL
      LHED(L) = LHED(L+1)
  132 CONTINUE
  136 CONTINUE
C
C     DETERMINE WHETHER TO ASSEMBLE,ELIMINATE,OR BACKSUBSTITUTE
C
      IF(LCOL.GT.NCRIT)GO TO 60
      IF(NELL.LT.NEL) RETURN
      IF(LCOL.GT.1)GO TO 60
      LCO = IABS(LHED(1))
      KPIVRO = 1
      PIVOT = EQ(1,1)
      KRO = LCO
      LPIVCO = 1
      QQ(1) = 1.
C     IF(NIT.EQ.0.OR.NPRA.EQ.0)GO TO 148
C     WRITE(NLP,452)LCO,KRO,PIVOT
      IF(ABS(PIVOT).LT.1D-08)GO TO 152
C 148 CONTINUE
      R1(KRO) = R1(KRO)/PIVOT
      WRITE(ND1) KRO,LCOL,LPIVCO,LHED(1),QQ(1)
C
C *** START BACK-SUBSTITUTION
C
      CALL BACSUB
     1    (NTOV ,NCOD ,BC    ,R1    ,DIS   ,MAXFR,QQ    ,LHED ,ND1   )

C *** MAIN EXIT WITH SOLUTION ******************************************
C
  152 CONTINUE
  417 FORMAT(/'  NERROR=',I5//
     1 '  THE DIFFERENCE NMAX-NCRIT IS NOT SUFFICIENTLY LARGE'
     1/'  TO PERMIT THE ASSEMBLY OF THE NEXT ELEMENT---'
     1/'  EITHER INCREASE NMAX OR LOWER NCRIT'
     1/)
  418 FORMAT(/'  NERROR=',I5//
     1 '  THERE ARE NO MORE ROWS FULLY SUMMED,THIS MAY BE DUE TO---'
     1/'  (1)INCORRECT CODING OF NOP OR NK ARRAYS'
     1/'  (2)INCORRECT VALUE OF NCRIT. INCREASE NCRIT TO PERMIT'
     1/'     WHOLE FRONT TO BE ASSEMBLED'
     1/)
C 452 FORMAT(13H PIVOTAL ROW=,I4,16H PIVOTAL COLUMN=,I4,7H PIVOT=,E20.10
C    1)
  476 FORMAT('  WARNING-MATRIX SINGULAR OR ILL CONDITIONED')
  484 FORMAT('  FRONTWIDTH VALUE=',I4)
      RETURN
      END
C
C *********************************************************************
C
      SUBROUTINE BACSUB
     1    (NTOTL,IFIX ,VFIX ,RHS  ,SOLN ,MFRNT,RWORK,IWORK,IDV2 )
C
C *********************************************************************
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION IFIX (NTOTL) ,VFIX (NTOTL) ,RHS  (NTOTL) ,SOLN (NTOTL)
      DIMENSION RWORK(MFRNT) ,IWORK(MFRNT)
C
C ***
C
C ***
                          DO 4990 IPOS=1,NTOTL
                          SOLN(IPOS)=0.0
                          IF(IFIX(IPOS).NE.0) SOLN(IPOS)=VFIX(IPOS)
 4990                     CONTINUE
C
      DO 5000 KPOS=1,NTOTL
C
      BACKSPACE IDV2
      READ(IDV2)     IPOS,IFRNT,JFRNT,(IWORK(K),RWORK(K),K=1,IFRNT)
      BACKSPACE IDV2
C
      IF(IFIX(IPOS).NE.0)  GO TO 5000
C
      WW            = 0.0
      RWORK(JFRNT)  = 0.0
C
                          DO 5010 K=1,IFRNT
                          JPOS=IABS(IWORK(K))
                          WW  =WW - RWORK(K)*SOLN(JPOS)
 5010                     CONTINUE
C
      SOLN(IPOS)=RHS(IPOS)+WW
 5000 CONTINUE
C
```

```
C ***
      RETURN
      END
C.......................................
C     PRE-FRONT ROUTINE                .
C.......................................
      SUBROUTINE PREFNT  ( NNM    , NEL    , NOP , NPE , MAXEL )
      DIMENSION NOP  ( MAXEL , 9 )
      NLAST = 0
      DO 12 I = 1,NNM
      DO 8  N = 1,NEL
C     JDN = NDN(N)
      DO 4 L =  1,NPE
      IF(NOP(N,L).NE.I)GO TO 4
      NLAST1 = N
      NLAST = N
      L1 = L
    4 CONTINUE
    8 CONTINUE
      IF(NLAST.EQ.0) GO TO 12
      NOP(NLAST,L1) = -NOP(NLAST,L1)
      NLAST = 0
   12 CONTINUE
      RETURN
      END
□
103


103


2


2
```