PETROV-GALERKIN FINITE ELEMENT MODEL FOR COMPRESSIBLE FLOWS USING ADAPTIVE REFINEMENT OF NONSTRUCTURED GRIDS

F.P. Brueckner

Publicación CIMNE Nº 10, Diciembre 1990

PETROV-GALERKIN FINITE ELEMENT MODEL FOR COMPRESSIBLE FLOWS USING ADAPTIVE REFINEMENT OF NONSTRUCTURED GRIDS

F.P. Brueckner

Department of Aerospace and Mechanical Engineering
The University of Arizona, Tucson

Publicación CIMNE Nº 10, Diciembre 1990

Centro Internacional de Métodos Numéricos en Ingeniería
Gran Capitán s/n, 08034 Barcelona, España

TABLE OF CONTENTS

SUMMARY

This report outlines the formulation and implementation of the Petrov-Galerkin finite element model under development at The University of Arizona into the adaptive computational mesh algorithms of the International Center for Numerical Methods in Engineering, Barcelona, Spain. This work included the formulation of linear triangular elements, their use on the unstructured grids generated by the mesh generator of the ICNME, and the subsequent numerical error estimation of the results and remeshing. The implementation yielded good results for the Petrov-Galerkin weighting of the convective terms. Additional algorithmic modifications were made in order to investigate the effects of alternate weighting schemes, higher order spatial and temporal integration schemes, more consistent treatment of the mass matrices, and a formulation based on the use of the internal energy. In general, oscillatory results were obtained for schemes in which all terms were weighted with the Petrov-Galerkin functions, but additional work in this area and an investigation of the use of the other modifications are needed. The computer codes generated or modified for use in this study are described herein. A copy of the flow solvers themselves is also included.

INTRODUCTION

Research is currently being performed at The University of Arizona on the development of a finite element model for the numerical approximation of the compressible Euler and Navier-Stokes equations. To date, this research has resulted in the formulation of Petrov-Galerkin weighting functions for these systems of equations and their implementation using bilinear quadrilaterial elements. The most attractive feature of this formulation is that precise criteria exist for the automatic determination of all parameters used, a feature in contrast to other schemes based on the addition of artificial viscosity. Early results have been encouraging with respect to the quality of the solutions obtained, the robustness of the algorithm, and the computational time required.

Concurrently, research at the International Center for Numerical Methods in Engineering at the Universitat Politècnica de Catalunya, Barcelona, Spain (ICNME), has been progressing on two fronts regarding the numerical approximation of these equations:

i. The use of numerical a posteriori error estimation and optimum adaptive refinement of nonstructured computational grids.

ii. The study of high- and low-order finite elements with emphasis on local stability in regions in which the flow is nearly incompressible.

An algorithm based on a two-step Taylor-Galerkin formulation has been used for the numerical simulations in this research. Here, an adaptation of the flux-corrected transport scheme of Boris and Book (1973) and Zalesak (1979) and an artificial viscosity of Lapidus type (Lapidus 1967) are employed in order to prevent oscillations and capture shocks. As a result, parameters are introduced for which no criteria exist for their numerical determination.

Because of the ability to uniquely determine optimal parameters within the Petrov-Galerkin formulation, it has been proposed that finite elements based on this method be incorporated into the work of the ICNME. This report describes the progress made between January 17 and February 9, 1990, toward this end. This includes the successful implementation of the Petrov-Galerkin formulation using three-noded linear elements and unstructured computational grids generated by both user-supplied mesh parameters and the resident error-estimating algorithm. In addition, the following items have been implemented into a computer code:

i. Petrov-Galerkin weighting
   - convective terms only
   - all terms (Euler only)
   - all terms with the discontinuity capturing of Hughes et al. (1986) (Euler only)

ii. Lumped or consistent mass matrices

iii. Local or global time stepping

iv. Inflow/outflow boundary conditions for the Euler equations based on the characteristics

v. Formulations based on the transport of total energy or internal energy

vi. First- or second-order Gaussian integration of triangular elements

The resulting semi-discrete equations are integrated in time using either a first-order Euler or second-order Runge-Kutta scheme.

Brief outlines of the formulations of the element equations and other key items are presented in the following sections. Also, the solutions of some example problems are given with a discussion of results and recommendations. Finally, a description of the computer programs and copies of the source codes are given.

GOVERNING EQUATIONS AND WEAK FORMULATION

Total Energy Form

The dimensionless governing equations in two dimensions are given by

$$\frac{\partial \underline{U}}{\partial t} + \frac{\partial \underline{F}_1}{\partial x} + \frac{\partial \underline{F}_2}{\partial y} = \frac{\partial \underline{G}_1}{\partial x} + \frac{\partial \underline{G}_2}{\partial y} \tag{1}$$

where

$$\underline{U} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{Bmatrix}, \quad \underline{F}_1 = \begin{Bmatrix} \rho u \\ \rho u^2 + \dfrac{p}{\gamma M_\infty^2} \\ \rho uv \\ \rho u E + (\gamma - 1) p u \end{Bmatrix}, \quad \underline{F}_2 = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + \dfrac{p}{\gamma M_\infty^2} \\ \rho v E + (\gamma - 1) p v \end{Bmatrix}$$

$$\underline{G}_1 = \begin{Bmatrix} 0 \\ \dfrac{1}{Re}\left[\dfrac{4}{3}\dfrac{\partial u}{\partial x} - \dfrac{2}{3}\dfrac{\partial v}{\partial y}\right] \\ \dfrac{1}{Re}\left[\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right] \\ (\gamma-1)\dfrac{\gamma M_\infty^2}{Re}\left[u\left(\dfrac{4}{3}\dfrac{\partial u}{\partial x} - \dfrac{2}{3}\dfrac{\partial v}{\partial y}\right) + v\left(\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right)\right] + \dfrac{\gamma}{PrRe}\dfrac{\partial T}{\partial x} \end{Bmatrix}$$

$$\underline{G}_2 = \begin{Bmatrix} 0 \\ \dfrac{1}{Re}\left[\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right] \\ \dfrac{1}{Re}\left[\dfrac{4}{3}\dfrac{\partial v}{\partial y} - \dfrac{2}{3}\dfrac{\partial u}{\partial x}\right] \\ (\gamma-1)\dfrac{\gamma M_\infty^2}{Re}\left[u\left(\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right) + v\left(\dfrac{4}{3}\dfrac{\partial v}{\partial y} - \dfrac{2}{3}\dfrac{\partial u}{\partial x}\right)\right] + \dfrac{\gamma}{PrRe}\dfrac{\partial T}{\partial y} \end{Bmatrix}$$

with the dimensionless equation of state

$$p = \rho T . \tag{2}$$

The equations were nondimensionalized using

$$p = \frac{\bar{p}}{p_\infty}, \quad x = \frac{\bar{x}}{L}, \quad y = \frac{\bar{y}}{L}, \quad \rho = \frac{\bar{\rho}}{\rho_\infty}, \quad T = \frac{\bar{T}}{T_\infty},$$

$$u = \frac{\bar{u}}{u_\infty}, \quad v = \frac{\bar{v}}{u_\infty}, \quad e = \frac{\bar{e}}{c_v T_\infty}, \quad \text{and} \quad t = \frac{\bar{t}}{L/u_\infty}$$

where overbars indicate dimensional quantities. Here, u and v are the x and y components, respectively; ρ is the density; p is the pressure; T is the temperature; e is the internal energy; E is the total energy [$\bar{E} = \bar{e} + 1/2\ (\bar{u}^2 + \bar{v}^2)$]; and t is the time. Also, γ is the ratio of specific heat: c_p/c_v; M_∞ is the free stream Mach number; Re is the Reynolds number, $\rho_\infty u_\infty L/\mu_\infty$; Pr is the Prandtl number, $\mu_\infty c_p/k$; μ_∞ is the coefficient of viscosity; and k is the thermal conductivity coefficient. The speed of sound is given by $\bar{a} = \sqrt{\gamma\bar{p}/\bar{\rho}}$. It may be shown that

$$E = e + \frac{\gamma(\gamma - 1)}{2} M_\infty^2 (u^2 + v^2) \tag{3}$$

and

$$T = e. \tag{4}$$

As a result, the equation of state (2) may be expressed as

$$p = \rho\left[E - \frac{\gamma(\gamma - 1)}{2} M_\infty^2 (u^2 + v^2)\right]. \tag{5}$$

Substituting (5) into (1), the following Petrov-Galerkin weak formulation may be obtained:

Continuity

$$\int_\Omega W_c\left[\frac{\partial\rho}{\partial t} + \rho\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \boxed{u\frac{\partial\rho}{\partial x} + v\frac{\partial\rho}{\partial y}}\right] d\Omega = 0 \tag{6a}$$

X-Momentum

$$
\begin{aligned}
&\int_{\Omega} \Bigg\{ W_u \Bigg\{ \rho \frac{\partial u}{\partial t} + \boxed{\rho \left[(2-\gamma) u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right]} \\
&\qquad + \frac{1}{\gamma M_{\infty}^{2}} = \left[e \frac{\partial \rho}{\partial x} + \rho \frac{\partial E}{\partial x} \right] - \rho(\gamma - 1) v \frac{\mu v}{\partial x} \Bigg\} \\
&\qquad + \frac{1}{Re} \left[\frac{\partial W_u}{\partial x} \left[\frac{4}{3} \frac{\partial u}{\partial x} - \frac{2}{3} \frac{\partial v}{\partial y} \right] + \frac{\partial W_u}{\partial y} \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] \right] \Bigg\} d\Omega \\
&\qquad - \int_{\Gamma} \left\{ \frac{W_u}{Re} \left[\left[\frac{4}{3} \frac{\partial u}{\partial x} - \frac{2}{3} \frac{\partial v}{\partial y} \right] n_x + \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] n_y \right] \right\} d\Gamma = 0
\end{aligned}
\tag{6b}
$$

Y-Momentum

$$
\begin{aligned}
&\int_{\Omega} \Bigg\{ W_v \Bigg\{ \rho \frac{\partial v}{\partial t} + \boxed{\rho \left[u \frac{\partial v}{\partial x} + (2-\gamma) v \frac{\partial v}{\partial y} \right]} \\
&\qquad + \frac{1}{\gamma M_{\infty}^{2}} \left[e \frac{\partial \rho}{\partial y} + \rho \frac{\partial E}{\partial y} \right] - \rho(\gamma - 1) u \frac{\partial u}{\partial y} \Bigg\} \\
&\qquad + \frac{1}{Re} \left[\frac{\partial W_v}{\partial x} \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] + \frac{\partial W_v}{\partial y} \left[\frac{4}{3} \frac{\partial v}{\partial y} - \frac{2}{3} \frac{\partial u}{\partial x} \right] \right] \Bigg\} d\Omega \\
&\qquad - \int_{\Gamma} \left\{ \frac{W_v}{Re} \left[\left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] n_x + \left[\frac{4}{3} \frac{\partial v}{\partial y} - \frac{2}{3} \frac{\partial u}{\partial x} \right] n_y \right] \right\} d\Gamma = 0
\end{aligned}
\tag{6c}
$$

Energy

$$\int_\Omega \Bigg\{ W_E \Bigg\{ \rho \frac{\partial E}{\partial t} + \boxed{\rho\left[\gamma_u \frac{\partial E}{\partial x} + \gamma_v \frac{\mu E}{\partial y}\right]}$$

$$+ (\gamma - 1) E\left[\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y}\right]$$

$$+ \frac{\gamma(\gamma-1)^2}{2} M_\infty^2 \left[\frac{\partial}{\partial x}(\rho u(u^2 + v^2)) + \frac{\partial}{\partial y}(\rho v(u^2 + v^2))\right]\Bigg\}$$

$$+ \frac{\gamma(\gamma-1)}{Re} M_\infty^2 \left[\frac{\partial W_E}{\partial x}\left\{u\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right) + v\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right\}\right.$$

$$\left. + \frac{\partial W_E}{\partial y}\left\{u\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + v\left(\frac{4}{3}\frac{\partial v}{\partial y} - \frac{2}{3}\frac{\partial u}{\partial x}\right)\right\}\right] \tag{6d}$$

$$+ \frac{\gamma}{PrRe}\left[\frac{\partial W_E}{\partial x}\frac{\partial T}{\partial x} + \frac{\partial W_E}{\partial y}\frac{\partial T}{\partial y}\right]\Bigg\} d\Omega$$

$$- \int_\Gamma \Bigg\{ W_E \frac{\gamma(\gamma-1)}{Re} M_\infty^2 \left[\left\{u\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right) + v\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right\} n_x\right.$$

$$\left. + \left\{u\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + v\left(\frac{4}{3}\frac{\partial v}{\partial y} - \frac{2}{3}\frac{\partial u}{\partial x}\right)\right\} n_y\right]$$

$$+ W_E \frac{\gamma}{PrRe}\left[\frac{\partial T}{\partial x} n_x + \frac{\partial T}{\partial y} n_y\right]\Bigg\} d\Gamma = 0 .$$

The discrete Petrov-Galerkin weighting functions used are given by

$$\begin{aligned} W_d(x, y) &= N^i(x, y) + P_c^i(x, y) \\ W_u(x, y) &= N^i(x, y) + P_u^i(x, y) \\ W_v(x, y) &= N^i(x, y) + P_v^i(x, y) \\ W_E(x, y) &= N^i(x, y) + P_E^i 1(x, y) \end{aligned} \tag{7}$$

where $N^i(x, y)$ is the ith shape function. The perturbation functions are as follows:

$$P_c^i(x, y) = \frac{h_c}{2|\underline{u}|}\left[u \frac{\partial N^i}{\partial x} + v \frac{\partial N^i}{\partial y}\right]$$

$$|\underline{u}| = (u^2 + v^2)^{1/2}$$

h_c = element length defined along (u, v)

$$P_u^i(x, y) = \frac{\alpha_u h_u}{2U}\left[(2 - \gamma)u \frac{\partial N^i}{\partial x} + v \frac{\partial N^i}{\partial y}\right]$$

$$U = \{[(2 - \gamma)u]^2 + v^2\}^{1/2}$$

$$\alpha_u = \coth \frac{\gamma_u}{2} + \frac{2}{\gamma_u}$$

$$\gamma_u = \frac{\rho Re}{1 + \frac{[(2 - \gamma)u]^2}{3U}} U h_u$$

h_u = element length defined along $[(2 - \gamma)u, v]$

$$P_v^i(x, y) = \frac{\alpha_v h_v}{2V}\left[u \frac{\partial N^i}{\partial x} + (2 - \gamma)v \frac{\partial N^i}{\partial y}\right]$$

$$V = \{u^2 + [(2 - \gamma)v]^2\}^{1/2}$$

$$\alpha_v = \coth \frac{\gamma_v}{2} + \frac{2}{\gamma_v}$$

$$\gamma_v = \frac{\rho Re}{1 + \frac{[(2 - \gamma)v]^2}{3V}} V h_v$$

h_v = element length defined along $[u, (2 - \gamma)v]$

$$P_E^i(x, y) = \frac{\alpha_E h_E}{2W} \left[\gamma u \frac{\partial N^i}{\partial x} + \gamma v \frac{\partial N^i}{\partial y} \right]$$

$$W = \gamma (u^2 + v^2)^{1/2}$$

$$\alpha_E = \coth \frac{\gamma_E}{2} + \frac{2}{\gamma_E}$$

$$\gamma_E = \frac{\rho RePr}{\gamma} Wh_E$$

h_E = element length defined along $(\gamma u, \gamma v)$

The algorithm defined by the finite element discretization of (6) is a consistent Petrov-Galerkin weighted residual formulation. The use of weighting functions defined by (7) in the convective terms only [boxed terms in (6)] and weighting all other terms by $N^i(x, y)$ (the standard Galerkin weighting functions) is equivalent to adding an anisotropic balancing diffusion, as explained by Kelly et al. (1980). It is this weighting of the convective terms that had been implemented prior to this work. As part of the investigation in Barcelona, an option was put in the computer code for using either weighting scheme. In addition, the discontinuity capturing term of Hughes et al. (1986) was also implemented as an option.

Internal Energy Form

It was seen in the previous section that the use of the equation describing the conservation of total energy introduces the kinetic energy into the momentum equations through the pressure gradient terms. It is this contribution that alters the directions and magnitudes of the convective velocities for the transport of u and v. As a result, a substantial amount of work is required to determine each weighting function.

It may be shown that the conservation of energy equation can be written alternatively in terms of the internal energy as

$$\rho\left[\frac{\partial e}{\partial t} + \underline{u} \cdot \nabla e\right] = \underline{\underline{\sigma}} : \nabla \underline{u} = \nabla \cdot \underline{q} \tag{8}$$

in which $\underline{\underline{\sigma}}$ is the stress tensor and $\underline{q}$ is the heat flux vector. Using the standard models for $\underline{\underline{\sigma}}$ and $\underline{q}$ and nondimensionalizing as described in the previous section, (8) may be expressed by

$$\rho\left[\frac{\partial e}{\partial t} + u\frac{\partial e}{\partial x} + v\frac{\partial e}{\partial y}\right] = -(\gamma - 1)p\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right]$$

$$+ \frac{\gamma(\gamma - 1)}{Re} M_\infty^2 \left\{\frac{4}{3}\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 - \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial y}\right)\right] + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + 2\left(\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}\right)\right\} \tag{9}$$

$$+ \frac{\gamma}{PrRe}\left[\frac{\partial^2 e}{\partial x^2} + \frac{\partial^2 e}{\partial y^2}\right].$$

Substituting the equation of state $p = \rho T = \rho e$ into the momentum equations and (9), the following Petrov-Galerkin weak formulation may be obtained:

<u>Continuity</u>

$$\int_\Omega W_c\left[\frac{\partial \rho}{\partial t} + \rho\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \boxed{u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y}}\right] d\Omega = 0 \tag{10a}$$

<u>X-Momentum</u>

$$\int_\Omega \left\{W_u\left[\rho\frac{\partial u}{\partial t} + \boxed{\rho\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right)} + \frac{1}{\gamma M_\infty^2}\frac{\partial(\rho e)}{\partial x}\right]\right.$$

$$\left. + \frac{1}{Re}\left[\frac{\partial W_u}{\partial x}\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right) + \frac{\partial W_u}{\partial y}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right]\right\} d\Omega \tag{10b}$$

$$- \int_\Gamma \left\{\frac{W_u}{Re}\left[\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right)n_x + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_y\right]\right\} d\Gamma = 0$$

Y-Momentum

$$\int_{\Omega}\left\{W_v\left[\rho\,\frac{\partial v}{\partial t}+\boxed{\rho\left(u\,\frac{\partial v}{\partial x}+v\,\frac{\partial v}{\partial y}\right)}+\frac{1}{\gamma M_{\infty}^{2}}\,\frac{\partial(\rho e)}{\partial y}\right]\right.$$

$$\left.+\frac{1}{Re}\left[\frac{\partial W_v}{\partial x}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)+\frac{\partial W_v}{\partial y}\left(\frac{4}{3}\,\frac{\partial v}{\partial y}-\frac{2}{3}\,\frac{\partial u}{\partial x}\right)\right]\right\}d\Omega \tag{10c}$$

$$-\int_{\Gamma}\left\{\frac{W_v}{Re}\left[\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)n_x+\left(\frac{4}{3}\,\frac{\partial v}{\partial y}-\frac{2}{3}\,\frac{\partial u}{\partial x}\right)n_y\right]\right\}d\Gamma=0$$

Energy

$$\int_{\Omega}\left\{W_E\left\{\rho\,\frac{\partial e}{\partial t}+\boxed{\rho\left(u\,\frac{\partial e}{\partial x}+v\,\frac{\partial e}{\partial y}\right)}+(\gamma-1)\rho\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)e\right.\right.$$

$$+\frac{\gamma(\gamma-1)}{Re}\,M_{\infty}^{2}\left[\frac{4}{3}\left(\left(\frac{\partial u}{\partial x}\right)^{2}+\left(\frac{\partial v}{\partial y}\right)^{2}-\left(\frac{\partial u}{\partial x}\,\frac{\partial v}{\partial y}\right)\right)\right.$$

$$\left.\left.+\left(\frac{\partial u}{\partial y}\right)^{2}+\left(\frac{\partial v}{\partial x}\right)^{2}+2\left(\frac{\partial u}{\partial y}\,\frac{\partial v}{\partial x}\right)\right]\right\} \tag{10d}$$

$$\left.+\frac{\gamma}{PrRe}\left[\frac{\partial W_E}{\partial x}\,\frac{\partial e}{\partial x}+\frac{\partial W_E}{\partial y}\,\frac{\partial e}{\partial y}\right]\right\}d\Omega$$

$$-\int_{\Gamma}\left\{W_E\,\frac{\gamma}{PrRe}\left[\frac{\partial e}{\partial x}\,n_x+\frac{\partial e}{\partial y}\,n_y\right]\right\}d\Gamma=0$$

The discrete Petrov-Galerkin weighting functions are given by

$$\begin{aligned}W_c(x,y)&=N^i(x,y)+P_c^i(x,y)\\W_u(x,y)&=N^i(x,y)+P_u^i(x,y)\\W_v(x,y)&=N^i(x,y)+P_v^i(x,y)\\W_e(x,y)&=N^i(x,y)+P_e^i(x,y)\end{aligned} \tag{11}$$

where $N^i(x, y)$ is the ith shape function. The perturbation functions are given by

$$P_c^i(x, y) = \frac{h}{2|\underline{u}|}\left[u\frac{\partial N^i}{\partial x} + v\frac{\partial N^i}{\partial y}\right]$$

$$P_u^i(x, y) = \frac{\alpha_u h}{2|\underline{u}|}\left[u\frac{\partial N^i}{\partial x} + v\frac{\partial N^i}{\partial y}\right]$$

$$\alpha_u = \coth\frac{\gamma_u}{2} + \frac{2}{\gamma_u}$$

$$\gamma_u = \frac{\rho Re}{1 + \frac{u^2}{3|\underline{u}|^2}}|\underline{u}|h$$

$$P_v^i(x, y) = \frac{\alpha_v h}{2|\underline{u}|}\left[u\frac{\partial N^i}{\partial x} + v\frac{\partial N^i}{\partial y}\right]$$

$$\alpha_v = \coth\frac{\gamma_v}{2} + \frac{2}{\gamma_v}$$

$$\gamma_v = \frac{\rho Re}{1 + \frac{v^2}{3|\underline{u}|^2}}|\underline{u}|h$$

$$P_e^i(x, y) = \frac{\alpha_e h}{2|\underline{u}|}\left[u\frac{\partial N^i}{\partial x} + v\frac{\partial N^i}{\partial y}\right]$$

$$\alpha_e = \coth\frac{\gamma_e}{2} + \frac{2}{\gamma_e}$$

$$\gamma_e = \frac{\rho Pr Re}{\gamma}|\underline{u}|h$$

The same comments made in the previous section regarding the weighting of the expressions derived are applicable.

Remarks

Several comments can be made with respect to the use of the energy equation as expressed in terms of the total energy versus the internal energy of the fluid:

i. The governing equations can no longer be written in conservative form when using the internal energy equations; i.e., in the form

$$\frac{\partial \underline{\phi}}{\partial t} + \frac{\partial \underline{F}(\underline{\phi})}{\partial \underline{x}} = \underline{0} .$$

ii. The proper directions for the application of the perturbation functions for the weighting of the equations are not the streamlines when using the total energy formulation. The presence of the kinetic energy results in different directions and/or magnitudes of the convective velocities for the transport of u, v, and E.

iii. The use of the internal energy equation results in the streamlines being the proper directions for upwinding, resulting in substantial computational savings over the formulation based on the total energy.

ALGORITHM IMPLEMENTATION

Triangular Elements and Numerical Integration

The dependent variables have been approximated by

$$\phi(\underline{x}, t) \cong \sum_i N^i(\underline{x})\phi_i(t)$$

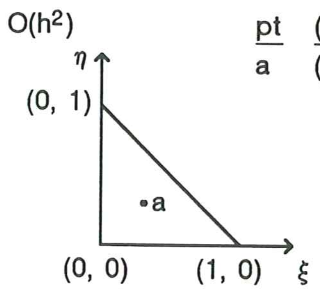
in which the shape functions used for the linear triangles are given by the following expressions in the local element coordinate system:

$$N^1(\xi, \eta) = 1 - \xi - \eta$$

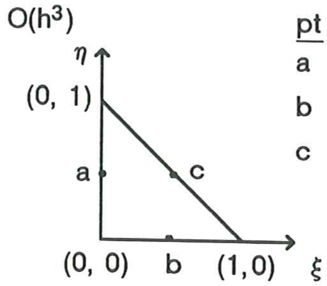
$$N^2(\xi, \eta) = \xi$$

$$N^3(\xi, \eta) = \eta .$$

The equations resulting from the substitution of these functions into the weak formulations of the preceding sections are integrated numerically in space according to the following (Zienkiewicz 1977):



pt	(ξ, η)	wt
a	$(\frac{1}{3}, \frac{1}{3})$	1



pt	(ξ, η)	wt
a	$(0, \frac{1}{2})$	$\frac{1}{3}$
b	$(\frac{1}{2}, 0)$	$\frac{1}{3}$
c	$(\frac{1}{2}, \frac{1}{2})$	$\frac{1}{3}$

An option has been included in the computer programs for the use of either of these integration rules

Solution of Equations

The semi-discrete Petrov-Galerkin formulations resulting from this spatial integration yield a system of equations of the form

$$\underline{\underline{M}}\,\frac{\partial \underline{\phi}}{\partial t} = \underline{R}$$

where

$\underline{\underline{M}}$ = mass matrix

$\underline{\phi}$ = solution vector

$\underline{R}$ = right-hand-side vector

Rather than invert the mass matrix, two alternative methods have been implemented.

i. *Method 1: Lumped Mass*

Here, the lumped mass matrix is defined by

$$\underline{\underline{M}}_\ell = M_{\ell ij}\hat{e}_i\hat{e}_j \quad \text{where } M_{\ell ij} = \begin{cases} \sum_j M_{ij} & i = j \\ 0 & i \neq j \end{cases}$$

and the inverse of the mass matrix is approximated as

$$\underline{\underline{M}}^{-1} \sim \underline{\underline{M}}_\ell^{-1} = M_{\ell ij}^{-1}\hat{e}_i\hat{e}_j \quad \text{where } M_{\ell ij}^{-1} = \begin{cases} \dfrac{1}{M_{\ell ij}} & i = j \\ 0 & i \neq j \end{cases}$$

The dependent variables are then obtained at the next time step by either of two integration schemes: a first-order Euler or a second-order Runge-Kutta.

ii. *Method 2: Consistent Mass*

The method used by Löhner et al. (1986) for the treatment of the mass matrix in a more consistent manner while maintaining the explicit nature of the algorithm has also been included. Here, a scheme has been developed in which the change in the dependent variables from one time step to the next is given by the following iterative expression:

$$\Delta\phi_0 = \Delta t \underline{\underline{M}}_\ell^{-1}\,\underline{R}$$

$$\Delta\phi_{r+1} = \Delta\phi_0 - \underline{\underline{M}}_\ell^{-1}(\underline{\underline{M}} - \underline{\underline{M}}_\ell)\Delta\phi_r \qquad r = 0, 1, \ldots$$

Definition of "h"

The perturbation terms of the Petrov-Galerkin weighting functions require the determination of an element length along the direction of convection of the dependent variable. This length has been calculated for quadrilateral elements in the manner explained by Yu and Heinrich (1987), for example. With the addition of triangular elements, the following definition of the element length has been employed:

$$h = \frac{|\underline{u}|}{|\underline{u}'|} h'$$

in which $|\underline{u}'|$ and $|\underline{u}|$ are the moduli of the velocity referenced to the local (element) and global coordinate systems, respectively. These velocities are related by

$$\underline{u}' = \underline{\underline{J}}^{-1} \underline{u}$$

where $\underline{\underline{J}}$ is the Jacobian matrix. The values of 2 and 1/2 have been implemented for quadrilateral and triangular elements, respectively, for the local length h'.

Time Step

For stability of the explicit time integration scheme used in this work, the following time-step limitations have been used:

Euler: $$\Delta t \leq \frac{h}{|\underline{u}| + a}$$

h = element length along $\underline{u}$

$|\underline{u}| = (u^2 + v^2)^{1/2}$

a = speed of sound = $\sqrt{\dfrac{\gamma p}{\rho}}$

Navier-Stokes: $$\Delta t \le \frac{1}{\frac{|\underline{u}| + a}{h_{\parallel}} + \frac{2k}{h_{\parallel}^2 + h_{\perp}^2}}$$

$$k = \frac{\gamma}{PrRe}$$

$h_{\parallel}$ = element length along $\underline{u}$

$h_{\perp}$ = element length perpendicular to $\underline{u}$

The time steps given by solving these expressions are computed for each element every ITIME time steps. Currently, ITIME = 10. Each node is assigned the smallest time step computed for its adjacent elements. Two types of time-stepping are employed:

Global: Smallest nodal Δt used everywhere.

Local: Nodal Δt used to advance each equation.

A safety factor S, defined by $\Delta t = S^*\Delta t$, has also been introduced and is read as input to the code.

Boundary Conditions

Euler

1. Inflow (from Usab and Murman 1985)

   i. $M \ge 1$: ρ, u, v, and e or E specified at infinity

   ii. $M < 1$: $q_{t_c} = q_{t_\infty}$

$$p_c = \frac{1}{2}[p_\infty + p_p + \bar{\rho}\,\bar{a}(q_{n_\infty} - q_{n_p})]$$

$$\rho_c = (p_c - p_\infty)/\bar{a}^2$$

$$q_{n_c} = q_{n_\infty} + (p_\infty - p_c)/(\bar{\rho}\,\bar{a})$$

where

q_t = tangent velocity

q_n = inward normal velocity

(p) = predicted state

(c) = corrected state

$(\bar{\cdot})$ = original state

The energy is computed from the pressure and density via the equation of state.

2. Outflow (from Usab and Murman 1985)

   i. $M \geq 1$: None

   ii. $M < 1$: $p_c = p_\infty$

$$\rho_c = \rho_p + (p_\infty - p_p)/\bar{a}^2$$

$$q_{t_c} = q_{t_p}$$

$$q_{n_c} = q_{n_p} + (p_\infty - p_p)/(\bar{\rho}\,\bar{a})$$

3. Solid Wall

   i. No condition on ρ or energy

   ii. $u_c = u_p(1 - n_x^2) - v_p\, n_x\, n_y$

$$v_c = v_p(1 - n_y^2) - u_p\, n_x\, n_y$$

Usab and Murman also have derived expressions for density and pressure at the wall, but these have not been introduced, as yet, into the codes.

Navier-Stokes

1. Inflow: All variables specified

2. Outflow: Zero natural boundary conditions, thereby satisfying

$$\underline{\underline{\tau}} \cdot \hat{n} = \underline{0}$$

$\underline{\underline{\tau}}$ = deviatoric stress tensor

$$\nabla T \cdot \hat{n} = 0$$

3. Solid Wall (Adiabatic): u and v specified

4. Solid Wall (Temperature Specified): u, v, and T specified

NUMERICAL EXAMPLES

The results of two examples are presented to illustrate the successful implementation of the Petrov-Galerkin formulation using linear triangular elements. The initial meshes for each example were generated with inputs supplied by the user. Subsequent meshes were the result of the error estimator based on the numerical solutions obtained. In each case, the internal energy formulation was used with first-order spatial and temporal integration schemes. In addition, the Petrov-Galerkin weights were only applied to the convective terms, and lumped mass matrices are used.

Example 1: Mach 5 Flow Over a Compression Corner

Mach 5 flow over a compression corner with a deflection angle of approximately 17 degrees was chosen for an initial comparison of results with those of the Taylor-Galerkin scheme. While this comparison has not as yet been performed, comparison with the analytical oblique shock solution indicates good agreement. The meshes employed and the resulting pressure contours are given in Figures 1 through 6.

Example 2: Mach 5 Flow With Compression and Expansion

The compression ramp geometry of the previous example was extended to include an expansion corner. The initial mesh and pressure contours are shown in Figures 7 and 8, respectively, and the final results are given in Figures 9 and 10. The improvement in the shock and expansion wave resolution is obvious.

Additional problems were attempted using some of the options described in the previous sections in order to verify their correct (or incorrect) implementation and their effect on the solutions. Also, both internal and external flow geometries were analyzed. A discussion of these will be given in the following section.

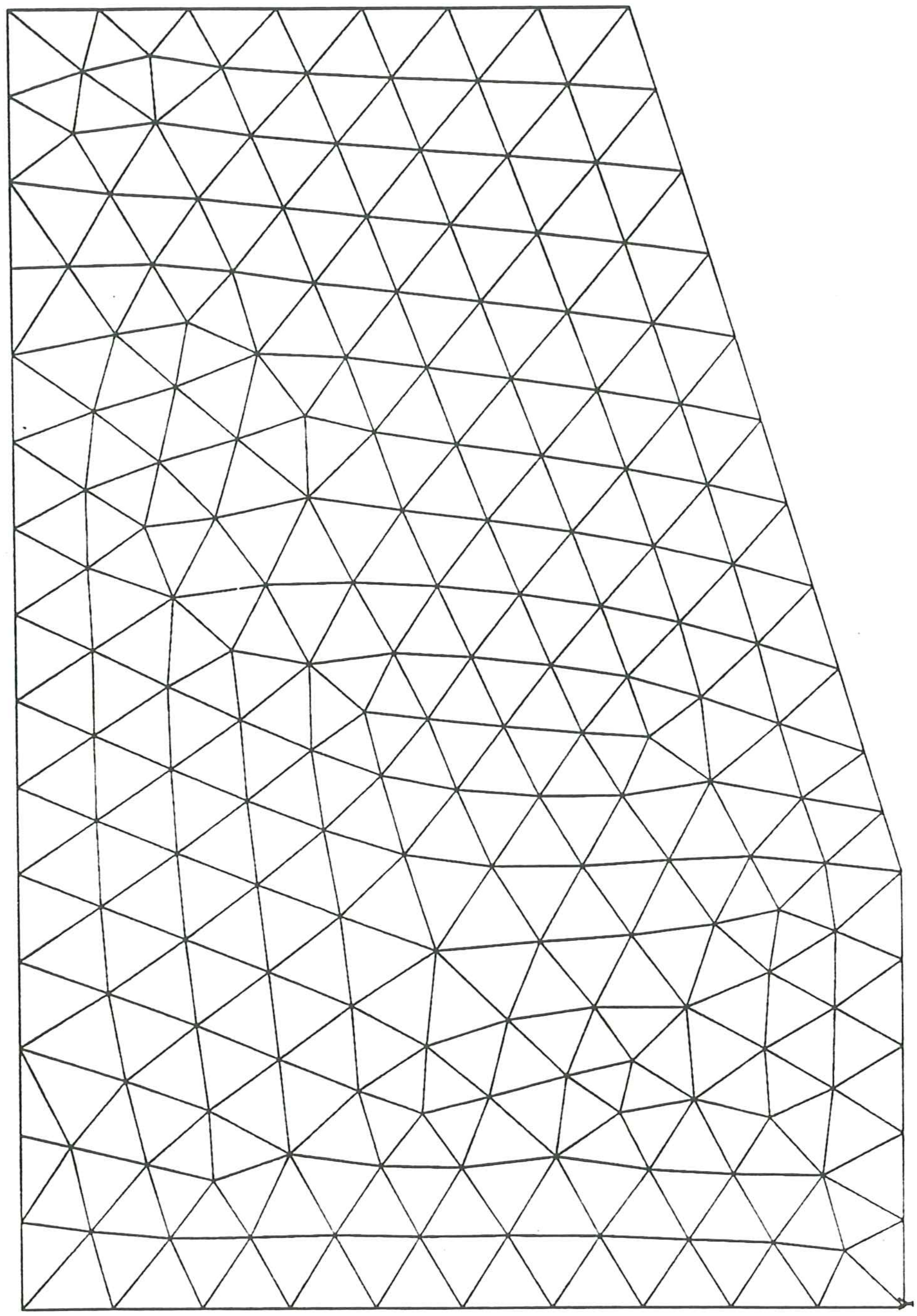


Figure 1. Initial mesh (example 1).

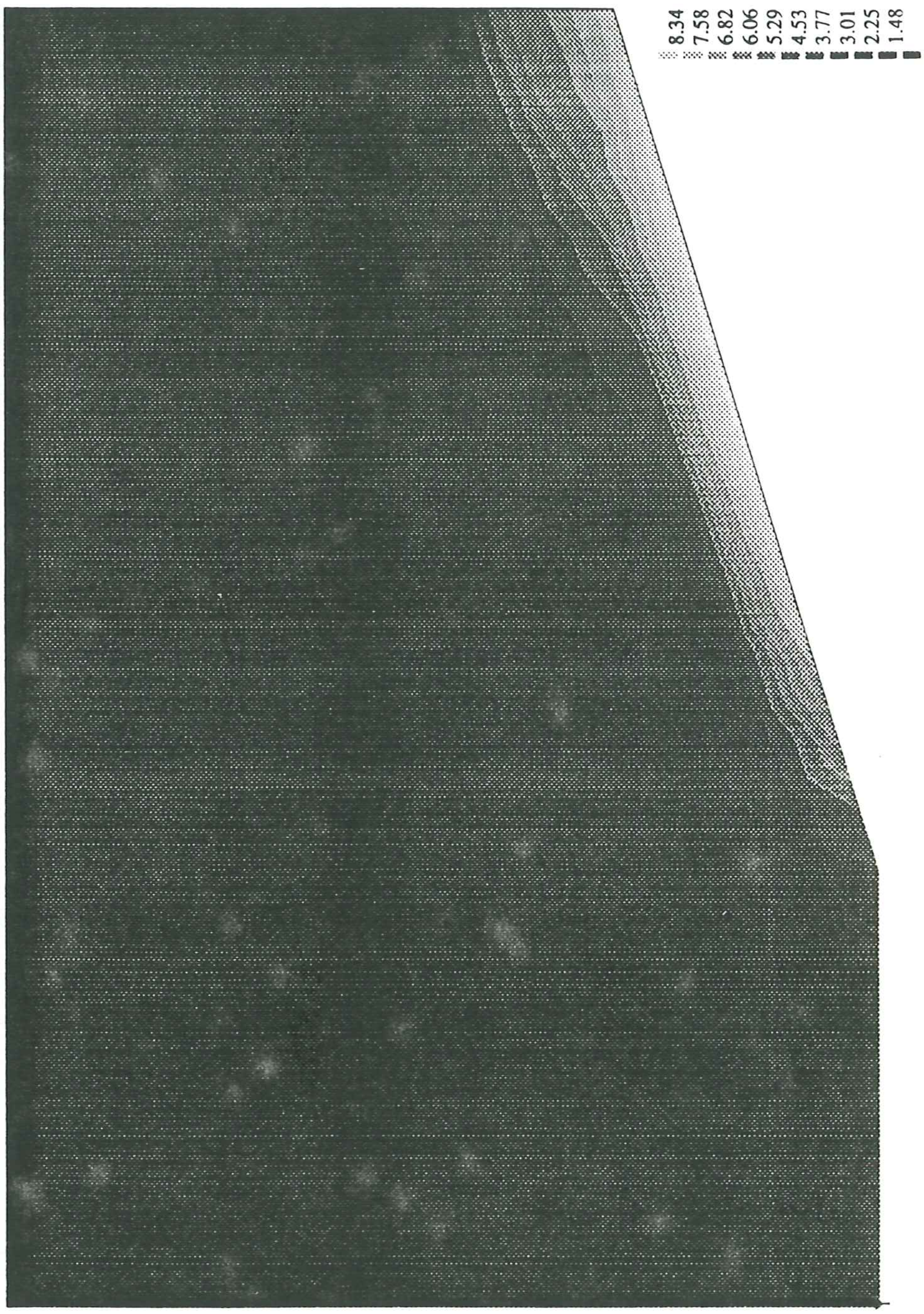


Figure 2. Pressure contours--initial mesh (example 1).

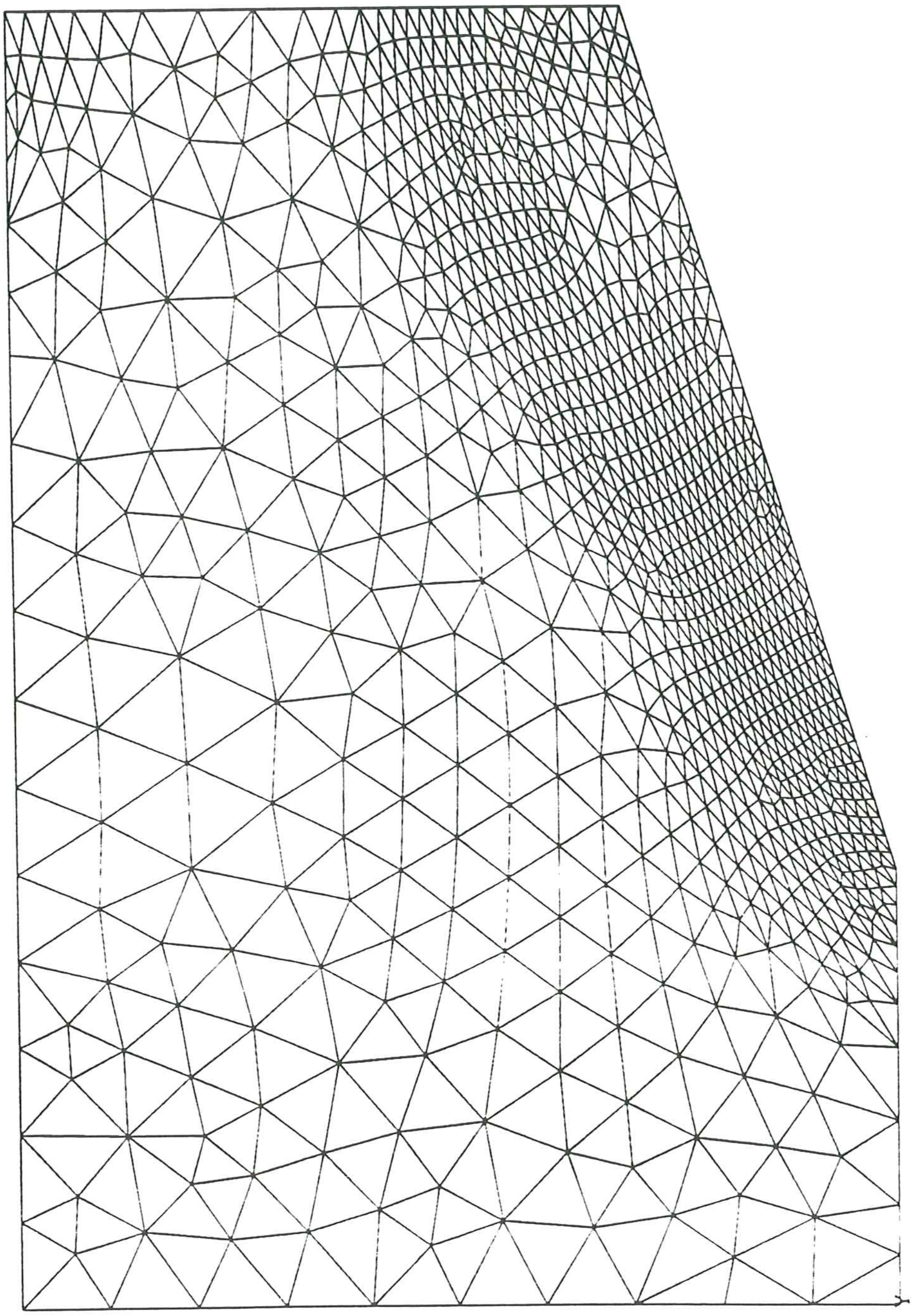


Figure 3. Second mesh (example 1).

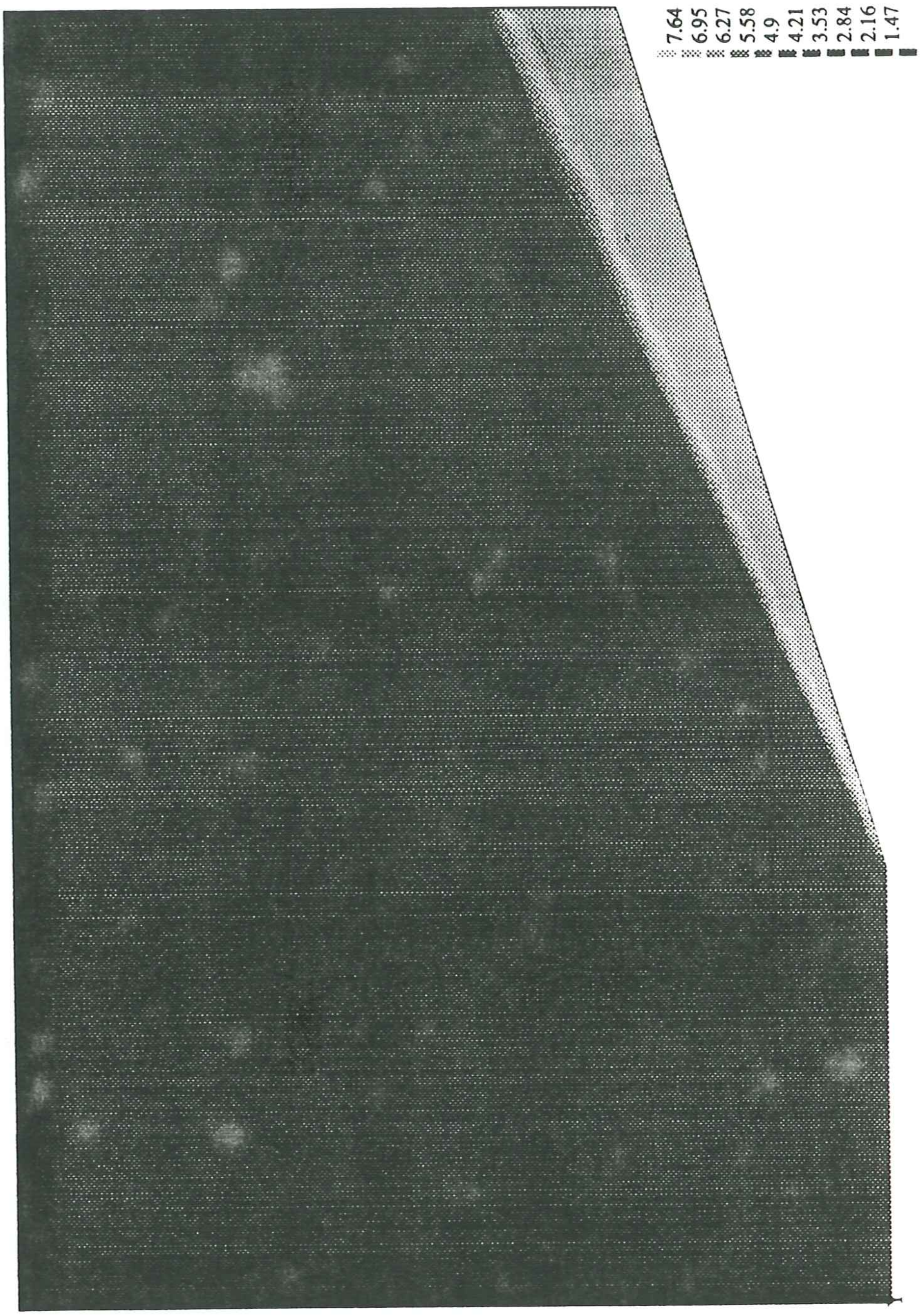


Figure 4. Pressure contours--second mesh (example 1).

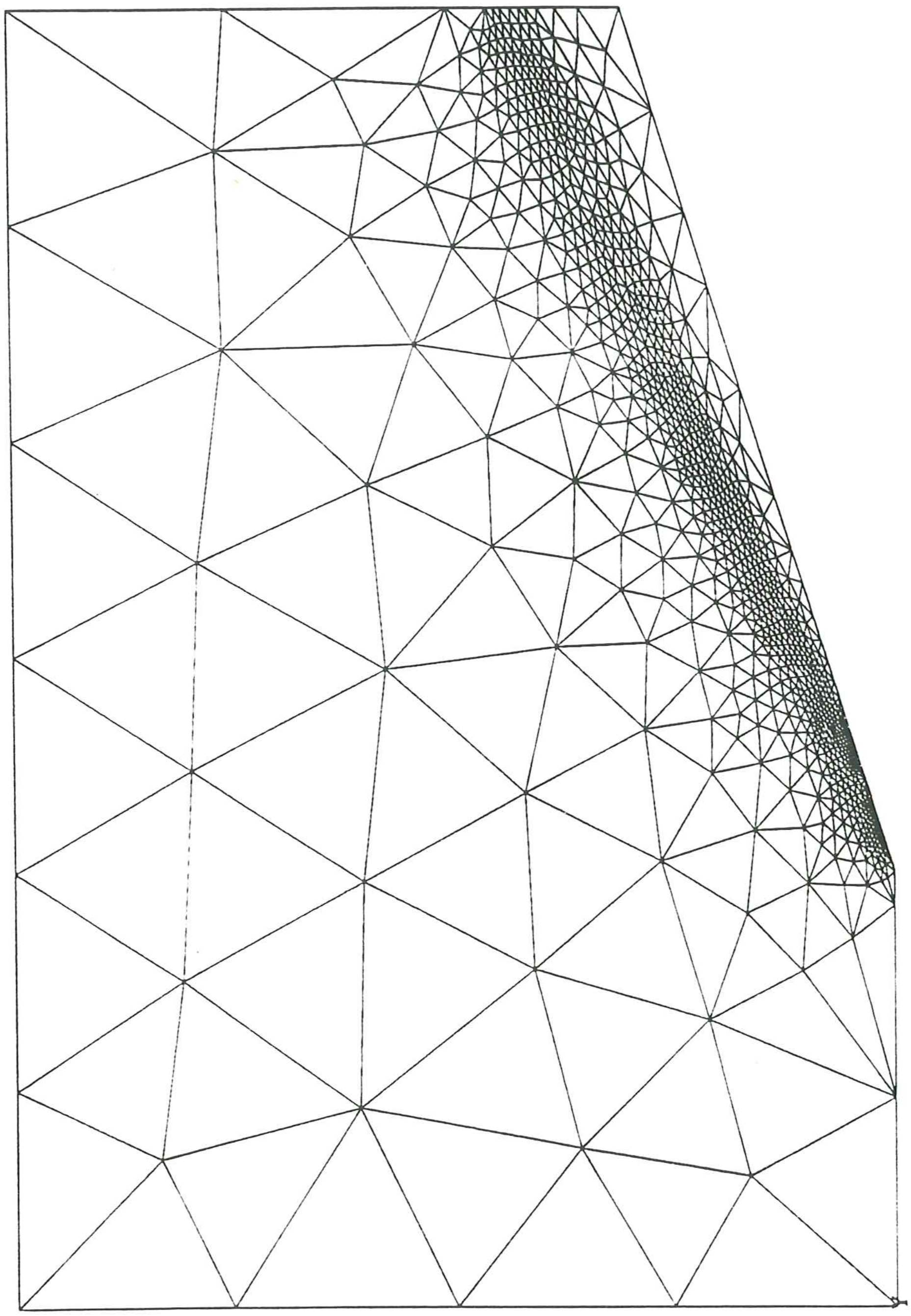


Figure 5. Final mesh (example 1).

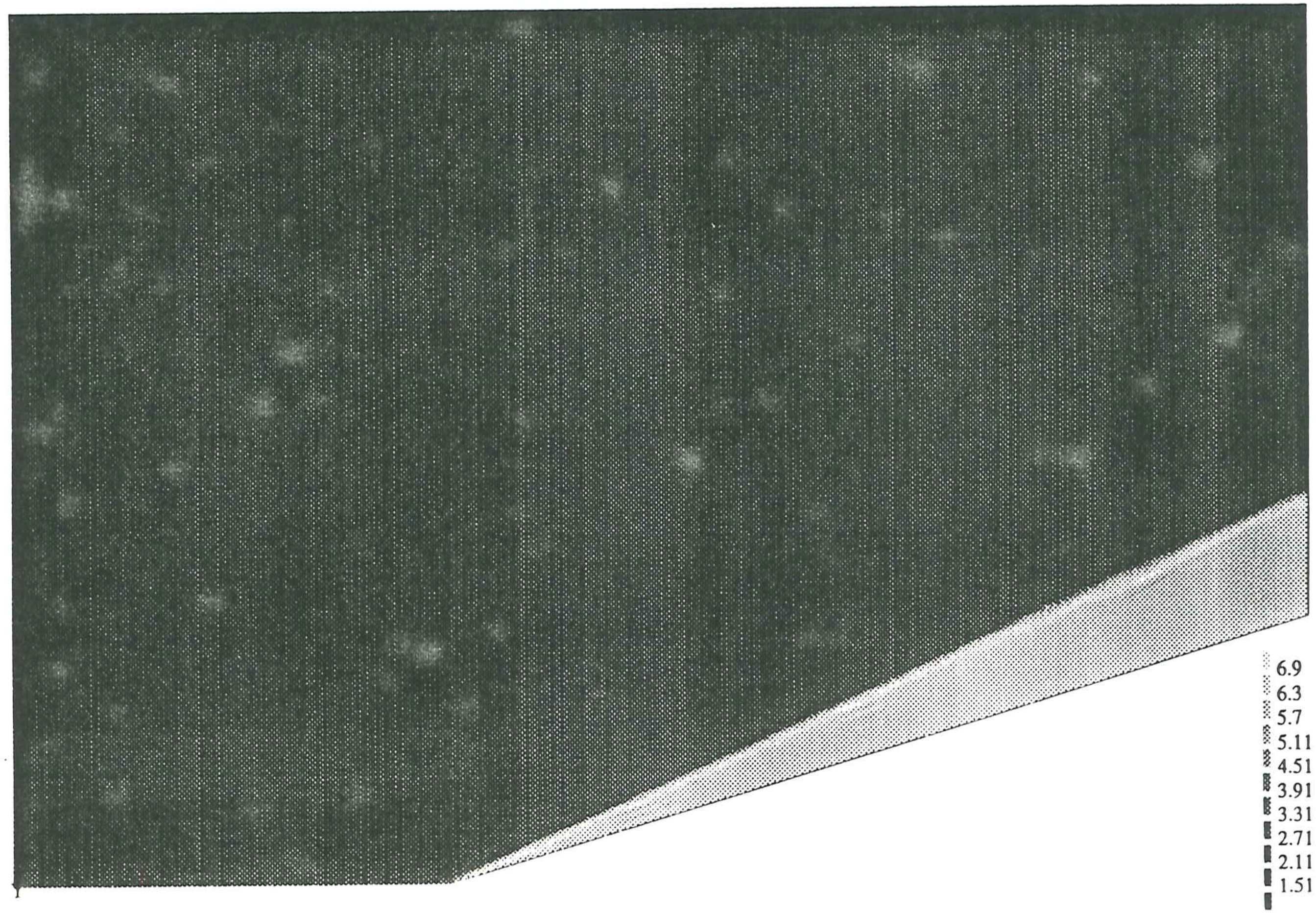


Figure 6. Pressure contours--final mesh (example 1).

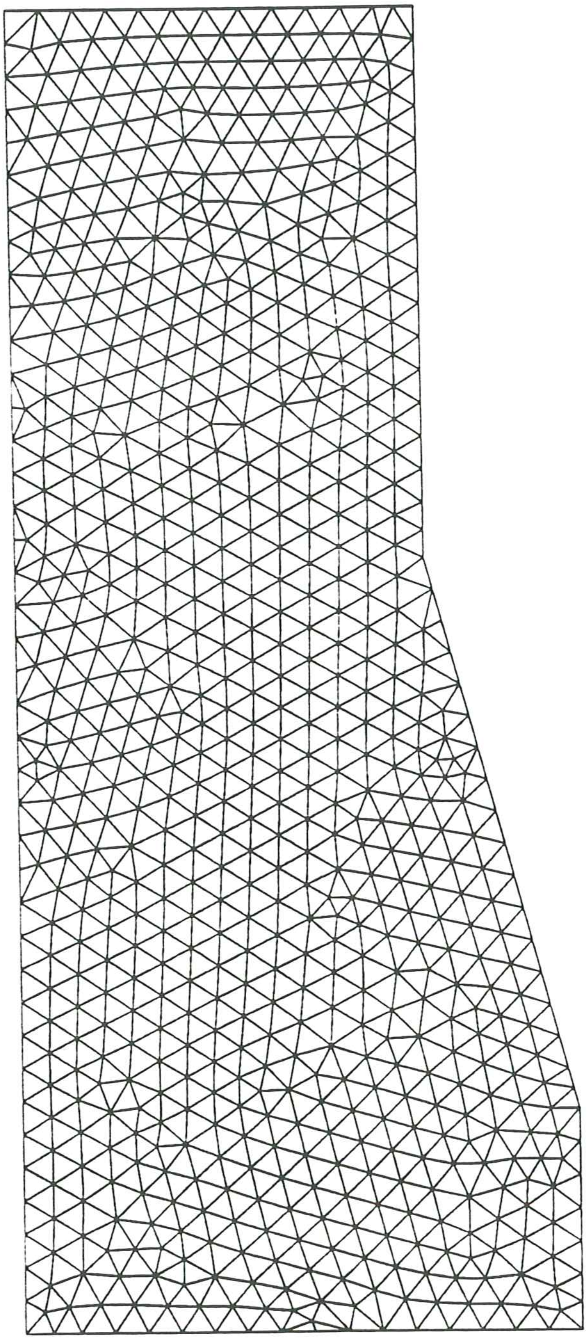


Figure 7. Initial mesh (example 2).

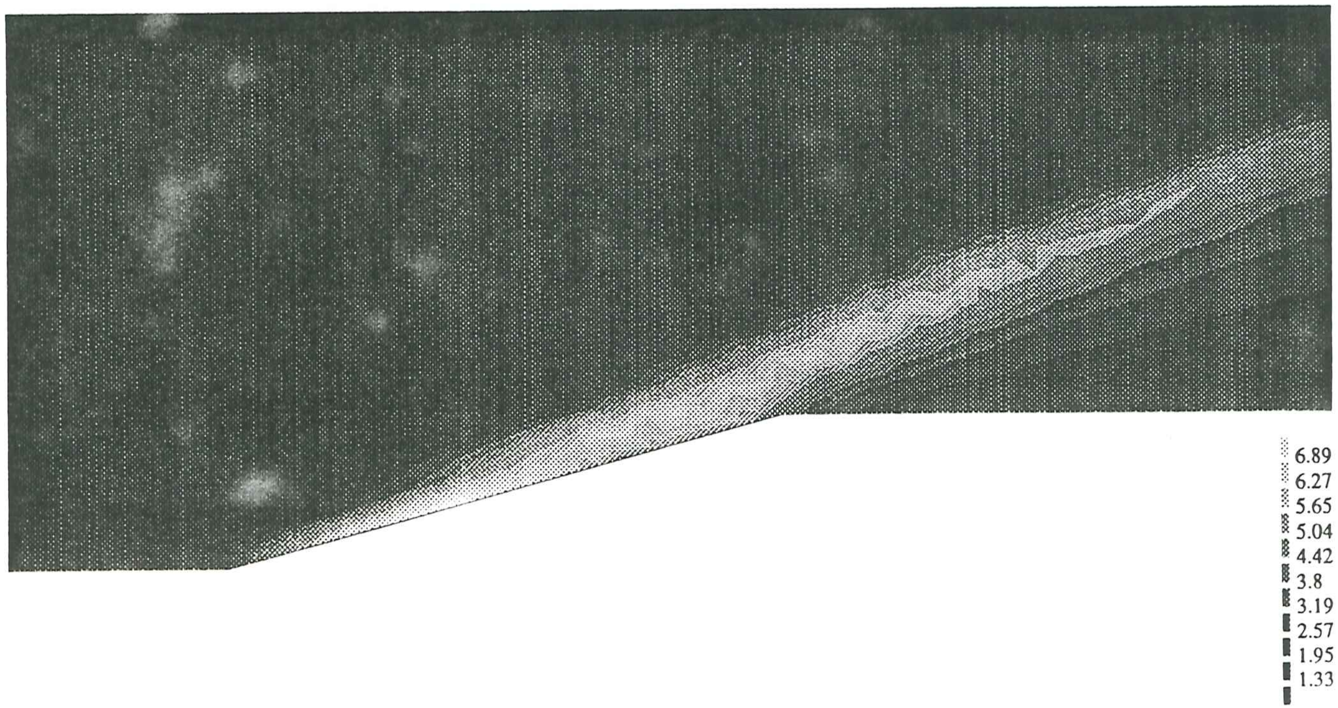


Figure 8. Pressure contours--initial mesh (example 2).

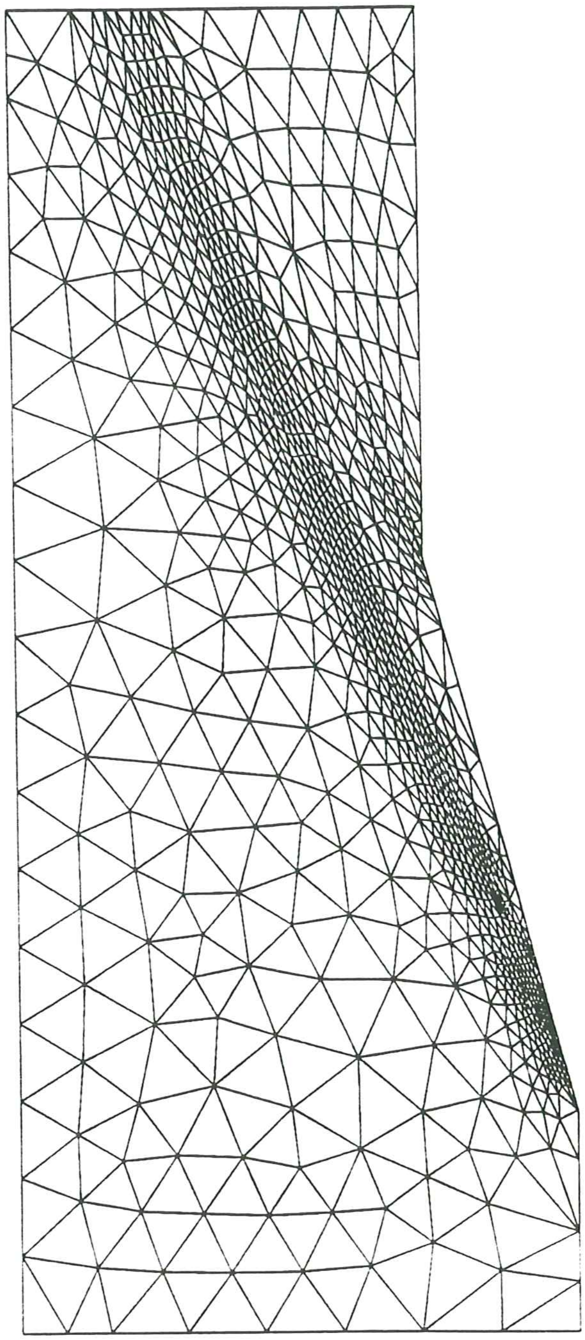


Figure 9. Final mesh (example 2).

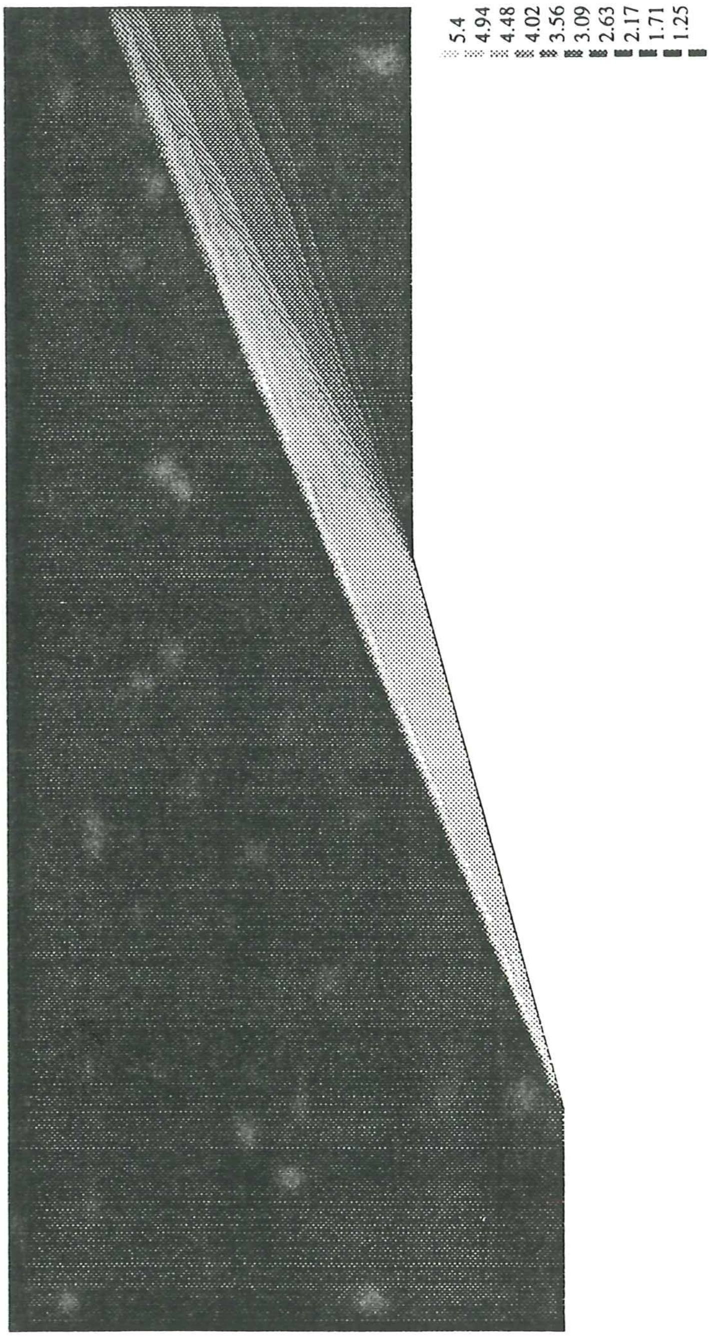


Figure 10. Pressure contours--final mesh (example 2).

DISCUSSION

The primary focus of this study has been to formulate the Petrov-Galerkin finite element model under development at The University of Arizona to triangular elements and to integrate the resulting algorithm into the adaptive remeshing codes of the ICNME in Barcelona. This work was, in fact, completed. The original scheme derived through the Petrov-Galerkin weighting of the convective terms was seen to yield results sufficiently good on extremely coarse meshes to obtain error estimates whose use yielded much improved meshes and, subsequently, numerical solutions. Final results were quite good, as illustrated by the solutions presented.

Based on the results of this work, additions to this algorithm were introduced in order to increase the accuracy of the scheme and make it adaptable to a more general use environment. In addition, an algorithm using the internal energy equation was also implemented and observed to yield comparable results to the original formulation while using a reduced amount of CPU time.

The use of the Petrov-Galerkin weighting functions was extended to all of the terms of the equations. The resulting scheme, consistent within a weighted residual formulation, yielded extremely sharp shocks (generally withon one element), but exhibited poor convergence behavior and oscillatory results downstream of the shocks. In an attempt to suppress these oscillations, the discontinuity capturing terms of Hughes et al. (1986) were added to the weighting functions. It is unclear at this point if the unsatisfactory results obtained with this algorithm are due to the formulation or its implementation.

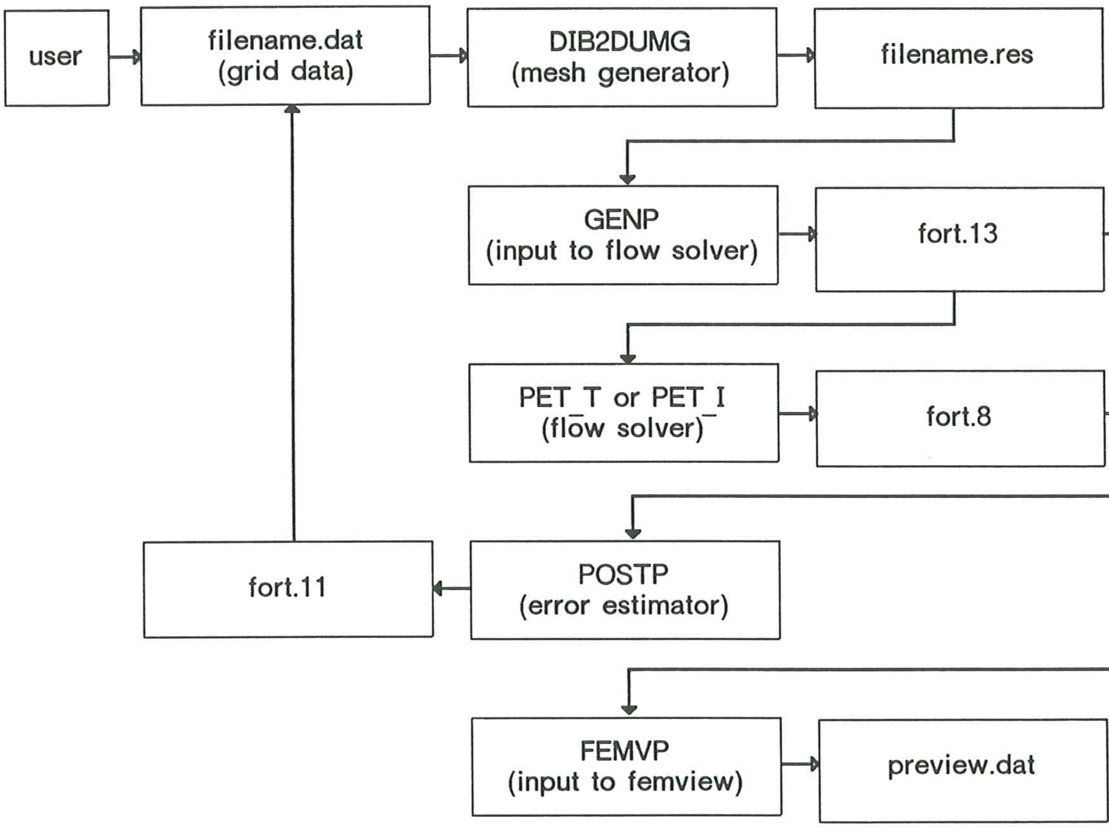
Several other modifications were also made in order to study their effect on the solutions and/or robustness of the algorithm. Higher order spatial and temporal integration schemes, consistent mass matrices, and local time-stepping were introduced for this purpose. Quantitative results of these schemes will be forthcoming. In general, for the steady-state results at this time, the solutions were all quite similar, regardless of the schemes used. The effects of the robustness of the flow solver itself, however, were not studied. The primary objective at this point was more on the implementation of these schemes than on a parametric type study due to time restrictions.

In addition, the inflow/outflow boundary conditions were changed to incorporate the conditions derived by Usab and Murman (1985) for the Euler equations. These conditions made it possible to attempt external flow problems. The examples run, however, indicated stability problems at the solid boundaries of the body. Converged solutions were only obtained for relatively low Mach numbers (< 3), and these were poor at the solid boundaries. It is not evident if the problem is one of initial conditions, boundary conditions, mesh density, locally incompressible flow, etc. The current implementation of the Euler wall boundary conditions is one obvious place to start looking.

As evident from the above discussion, a substantial amount of work lies ahead with respect to the development of a high-speed flow software package based on Petrov-Galerkin methods and adaptive computational grids. In general, though, it is believed that with the work described in this report, and the quality of the results obtained, significant progress has been made toward this goal.

PROGRAM GUIDE

Two programs have been written for the solution of the compressible Euler and Navier-Stokes equations; one, PET_T.F, has been based on the use of the total energy formulation, while the other, PET_I.F, has been based on the use of the internal energy. The input and output of each code is the same. The source codes are included as appendices to this report. In addition, FORTRAN programs have been written as interfaces to the mesh generator, the error estimator, and FEMVIEW. A flow chart of this set of programs may be sketched as follows:



<u>Program Descriptions</u>

GENP.F

This program generates input data for the flow solver from the output of the mesh generator and some user inputs. Inputs are entered interactively, when asked, upon program execution as follows:

"Enter filename with mesh data" (Enter the name of the file created by the mesh generator)

The remaining inputs will be described in detail during the description of the flow solvers. File fort.13 is created for direct input to either flow solver.

PET_I.F and PET_T.F.

As mentioned earlier, the input and output data required or created for each flow solver are identical. The input is as follows and is read unformatted unless specified:

1. Title (A60) [If the mesh generator and GENP have been used, the title is carried along from there.]
2. NELTYPE (Integer)

	1:	bilinear quadrilaterals
	2:	linear triangles, one-point integration
	3:	linear triangles, three-point integration

3. NN, NE, NB (Integers)

NN:	number of nodes
NE:	number of elements
NB:	number of boundary conditions

4. MUPW, NFLOW, NCON, NTST, ITYPE, ISTART (Integers)

MUPW	1:	upwind convective terms only
	2:	upwind all terms
	3:	upwind all terms with discontinuity capturing
NFLOW	1:	Euler time integration
	2:	Runge-Kutta time integration
NCON	1:	lumped mass
	>1:	number of iterations for consistent mass (usually 3)
NTST	1:	local time-stepping
	2:	global time-stepping

ITYPE 0: Navier-Stokes solver

1: Euler solver

ISTART 0: initial conditions set at free stream values

1: restart; output from previous run must be put in file fort.4

5. XMACH, RE, PR, GAMMA, ALF (Real)

XMACH: free stream Mach number

RE: Reynolds number

PR: Prandtl number

GAMMA: ratio of specific heats

ALF: angle of attack

6. SAFE (Real) [time step safety factor, Δt = SAFE Δt]

7. NUM, (COOR(NUM,I), I=1,2) (Integer and 2 Real) [total of NN lines]

NUM: node number

COOR(NUM,I): x and y coordinates for i = 1, 2, respectively

8. NUM, (NO(NUM,I), I=1,NNODE) (Integers) [total of NE lines)

NUM: element number

NO(NUM,I): node numbers (element connectivity) NNODE is set to 4 for quadrilaterals and 3 for triangles

9. NUM,(IBOUND(NUM,I) ,I=1,4) (Integers) [total of NB lines)

NUM: boundary condition number

IBOUND(NUM,1): first node on boundary

IBOUND(NUM,2): second node on boundary

IBOUND(NUM,3): element number

IBOUND(NUM,4): type of boundary 1: free (inflow/outflow)

2: adiabatic wall

3: specified temperature wall

4: symmetry

File fort.8 is created for output. It includes the title, the number of elapsed time steps, and nodal results in the following order: node number, density, x-velocity, y-velocity, internal energy, total energy, and local Mach number. In addition, the residuals for the equations of each variable and the total are output to fort.6 every 10 time steps. These residuals are defined as the square of the L_2 norm to the right-hand sides of the subject equations.

POSTP.F

This program is a modification of the resident error estimator which uses as input the input file for and output file from the flow solver. The output of this program is a new background mesh for the mesh generator.

FEMVP.F

This program uses the input to the flow solvers and the output created by the flow solvers to generate file preview.dat for FEMVIEW.

ACKNOWLEDGMENTS

I would like to acknowledge the very helpful and supportive researchers of the ICNME for making my time in Barcelona both very productive and enjoyable. In particular, I would like to thank Ramon Codina, Fernando Quintana, and Gilbert Peffer for their help and insights and Dominic Clark for making sure I never ran out of disk space. Finally, I would like to thank Professors Eugenio Oñate and Juan Heinrich for giving me this marvelous opportunity.

LITERATURE CITED

Boris, J. P. and Book, D. L. 1973. "Flux Corrected Transport; I. SHASTA, A Transport Algorithm That Works," *J. Comp. Phys.*, Vol. 11, pp. 38-69.

Hughes, T. J. R., Mallet, M., and Mizukami, A. 1986. "A New Finite Element Formulation for Computational Fluid Dynamics; II. Beyond SUPG," *Comp. Meth. Appl. Mech. Engr.*, Vol. 54, pp. 341-355.

Kelly, D. W., Nakazawa, S., Zienkiewicz, O. C., and Heinrich, J. C. 1980. "A Note on Upwinding and Anisotropic Balancing Dissipation in Finite Element Approximations to Convection Diffusion Problems," *Int. J. Num. Meth. Engng.*, Vol. 15, pp. 1705-1711.

Lapidus, A. 1967. "A Detached Shock Calculation by Second Order Finite Differences," *J. Comp. Phys.*, Vol. 2, pp. 154-177.

Löhner, R., Morgan, K., Vahdati, M., Boris, J. P., and Book, D. L. 1986. "FEM-FCT: Combining Unstructured Grids With High Resolution," Report C/R/539/86, University College of Swansea, Swansea, United Kingdom.

Usab, W. J., Jr., and Murman, E. M. 1985. "Embedded Mesh Solutions of the Euler Equations Using a Multiple-Grid Method," in *Advances in Computational Transonics* (edited by W. G. Habashi), Pineridge Press, Swansea, United Kingdom.

Yu, C.-C. and Heinrich, J. C. 1987. "Petrov-Galerkin Methods for Multidimensional, Time-Dependent, Convective-Diffusion Equations," *Int. J. Num. Meth. Engng.*, Vol. 24, pp. 2201-2215.

Zalesak, S. T. 1979. "Fully Multidimensional Flux-Corrected Transport Algorithm for Fluids," *J. Comp. Phys.*, Vol. 31, pp. 335-362.

Zienkiewicz, O. C. 1977. *The Finite Element Method*, 3rd Edition, McGraw-Hill, London.

APPENDIX I

PROGRAM PET_I.F.

```
      PROGRAM PET_I
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  SOLUTION OF TWO-DIMENSIONAL COMPRESSIBLE FLOW BY A PETROV-
C  GALERKIN FINITE ELEMENT METHOD.
C
C
C  FIRST OR SECOND ORDER TIME INTEGRATION
C
C  ELEMENTS: (1) BILINEAR QUADRILATERALS
C            (2) LINEAR/LINEAR TRIANGLES WITH ONE POINT INTEGRATION
C            (3) LINEAR/LINEAR TRIANGLES WITH THREE POINT INTEGRATION
C
C  PETROV-GALERKIN WEIGHTING OF ALL OR ONLY CONVECTIVE TERMS
C
C  INTERNAL ENERGY FORMULATION
C
C  LOCAL OR GLOBAL TIMESTEPPING
C
C  LUMPED OR CONSISTENT MASS MATRICES
C
C  WITH OR WITHOUT DISCONTINUITY CAPTURING
C
C  INFLOW/OUTFLOW EULER BOUNDARY CONDITIONS PER USAB AND MURMAN
C
C
C                    FRANK P. BRUECKNER
C                    UNIVERSITY OF ARIZONA
C                    FEBRUARY 6, 1990
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
C  DEFINE COMMON BLOCKS
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2
C
      COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
      CHARACTER*60 INFILE, OUTFILE, TITLE
C
      DIMENSION DELT(NDNODE), TIME(NDNODE),
     1 UNEW(NDNODE), VNEW(NDNODE), RHONEW(NDNODE), ENEW(NDNODE),
     2 UHALF(NDNODE), VHALF(NDNODE), RHOHALF(NDNODE), EHALF(NDNODE),
     3 UOLD(NDNODE), VOLD(NDNODE), RHOOLD(NDNODE), EOLD(NDNODE)
C
      DATA NEXT /0/
C
C  INITIALIZE SOME VARIABLES
C
```

```
      IST=10
      ITI=10
      ISC=100
      ITOTAL=1
      ISTEADY=1
      ITIME=1
      ISCRAT=1
      TTOTAL=0.
      CONTOT=1.D3
C
C  READ CONTROL DATA
C
      READ(5,1) TITLE
      READ(5,*) NELTYPE
      READ(5,*) NN, NE, NBOUN
      WRITE(6,1) TITLE
      WRITE(6,*) ' '
C
C  SET ELEMENT PARAMETERS
C
      IF (NELTYPE .EQ. 1) THEN
        NGAUSS=4
        NNODE=4
        F=2.D0
      END IF
      IF (NELTYPE .EQ. 2) THEN
        NGAUSS=1
        NNODE=3
        F=.5D0
      END IF
      IF (NELTYPE .EQ. 3) THEN
        NGAUSS=3
        NNODE=3
        F=.5D0
      END IF
C
C  READ GLOBAL DATA AND SET INITIAL CONDITIONS
C
      CALL DINPT(NN,NE,NBOUN,RHONEW,UNEW,VNEW,ENEW,
     1            RHOOLD,UOLD,VOLD,EOLD,ITOTAL)
C
C  CALCULATE SOME CONSTANTS
C
      XM2=XMACH*XMACH
      GM1=GAMMA-1.D0
      ONEDGM2=1.D0/(GAMMA*XM2)
      C1=GAMMA*GM1*XM2/RE
      C2=GAMMA/(PR*RE)
C
C  COMPUTE BOUNDARY INFORMATION
C
      CALL BINFO(NN,NBOUN,NB,
     1 UNEW, VNEW, RHONEW, ENEW, UOLD, VOLD, RHOOLD, EOLD)
C
C  CALCULATE SHAPE FUNCTION AND ELEMENT DATA
C
      CALL ELEMENT(NE)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C                          NAVIER-STOKES SOLVER
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      IF (ITYPE .EQ. 1) GO TO 140
C
```

```
C  CALCULATE TIME STEP
C
         CALL TSTEP1(NN,NE,UNEW,VNEW,ENEW,TIME,F)
   40    IF (ITIME .EQ. ITI) THEN
           CALL TSTEP1(NN,NE,UNEW,VNEW,ENEW,TIME,F)
           ITIME=0
         END IF
C
C  CALCULATE AND ASSEMBLE GLOBAL EQUATIONS
C
         CALL ASSEM1(NN,NE,RHOOLD,UOLD,VOLD,EOLD,F)
         IF (NFLOW .EQ. 1) GO TO 41
C
C  ADVANCE TO TIME T + .5 * DELTA T FOR RUNGE-KUTTA INTEGRATION
C
         DO 42 I=1,NN
   42      DELT(I)=TIME(I)*.5D0
         CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1              RHOHALF,UHALF,VHALF,EHALF,DELT)
         CALL ASSEM1(NN,NE,RHOHALF,UHALF,VHALF,EHALF,F)
C
C  ADVANCE INT TIME TO T + DELTA T
C
   41    CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1              RHONEW,UNEW,VNEW,ENEW,TIME)
         TTOTAL=TTOTAL+TIME(1)
C
C  OUTPUT RESULTS TO RESTART FILE
C
         IF (ISCRAT .EQ. ISC) THEN
           CALL DOUTPT(NN,TITLE,ITOTAL,RHONEW,UNEW,VNEW,ENEW)
           ISCRAT=0
         END IF
C
C  CHECK FOR STEADY STATE
C
         IF (ISTEADY .EQ. IST) THEN
           CALL STEADY(NN,ITOTAL,NEXT,CONTOT)
           ISTEADY=0
           IF (NEXT .EQ. 1) GO TO 1000
         END IF
C
C  REASSIGN VARIABLES
C
         DO 45 I=1,NN
           RHOOLD(I)=RHONEW(I)
           UOLD(I)=UNEW(I)
           VOLD(I)=VNEW(I)
           EOLD(I)=ENEW(I)
   45    CONTINUE
         ITOTAL=ITOTAL+1
         ISTEADY=ISTEADY+1
         ITIME=ITIME+1
         ISCRAT=ISCRAT+1
         IF (ITOTAL .GT. 10000) STOP
         GO TO 40
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C                       EULER SOLVER
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C
C  CALCULATE TIME STEP
C
```

```
 140    CALL TSTEP2(NN,NE,UNEW,VNEW,ENEW,TIME,F)
 150    IF (ITIME .EQ. ITI) THEN
          CALL TSTEP2(NN,NE,UNEW,VNEW,ENEW,TIME,F)
          ITIME=0
        END IF
C
C  CALCULATE AND ASSEMBLE GLOBAL EQUATIONS
C
        CALL ASSEM2(NN,NE,RHOOLD,UOLD,VOLD,EOLD,F)
        IF (NFLOW .EQ. 1) GO TO 141
        DO 142 I=1,NN
 142      DELT(I)=TIME(I)*.5D0
C
C  ADVANCE TO TIME T + .5 * DELTA T FOR RUNGE-KUTTA INTEGRATION
C
        CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1             RHOHALF,UHALF,VHALF,EHALF,DELT)
        CALL ASSEM2(NN,NE,RHOHALF,UHALF,VHALF,EHALF,F)
C
C  ADVANCE TO TIME T + DELTA T
C
 141    CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1             RHONEW,UNEW,VNEW,ENEW,TIME)
        TTOTAL=TTOTAL+TIME(1)
C
C  OUTPUT RESULTS TO RESTART FILE
C
        IF (ISCRAT .EQ. ISC) THEN
          CALL DOUTPT(NN,TITLE,ITOTAL,RHONEW,UNEW,VNEW,ENEW)
          ISCRAT=0
        END IF
C
C  CHECK FOR STEADY STATE
C
        IF (ISTEADY .EQ. IST) THEN
          CALL STEADY(NN,ITOTAL,NEXT,CONTOT)
          ISTEADY=0
          IF (NEXT .EQ. 1) GO TO 1000
        END IF
C
C  REASSIGN VARIABLES
C
        DO 145 I=1,NN
          RHOOLD(I)=RHONEW(I)
          UOLD(I)=UNEW(I)
          VOLD(I)=VNEW(I)
          EOLD(I)=ENEW(I)
 145    CONTINUE
        ITOTAL=ITOTAL+1
        ISTEADY=ISTEADY+1
        ISCRAT=ISCRAT+1
        ITIME=ITIME+1
        IF (ITOTAL .GT. 8000) STOP
        GO TO 150
C
C  OUTPUT FINAL RESULTS
C
1000    CALL DOUTPT(NN,TITLE,ITOTAL,RHONEW,UNEW,VNEW,ENEW)
        STOP
C
C  FORMATS
C
   1    FORMAT (A60)
        END
C
C
```

```
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE DINPT(NN,NE,NBOUN,RHO1,U1,V1,E1,
     1                 RHO2,U2,V2,E2,ITOTAL)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  THIS SUBROUTINE READS IN THE GLOBAL DATA AND SETS
C  DEPENDENT VARIABLES TO INITIAL CONDITIONS
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
      DIMENSION RHO1(NDNODE), U1(NDNODE), V1(NDNODE), E1(NDNODE),
     1 RHO2(NDNODE), U2(NDNODE), V2(NDNODE), E2(NDNODE)
C
      CHARACTER*60 TEXT
C
C
C  READ PROGRAM OPTIONS
C
      READ (5,*) MUPW, NFLOW, NCON, NTST, ITYPE, ISTART
C
C  READ FREE STREAM PARAMETERS
C
      READ (5,*) XMACH, RE, PR, GAMMA, ALF
C
C  READ TIME STEP SAFETY FACTOR
C
      READ (5,*) SAFE
C
C  READ NODAL COORDINATES
C
      DO 10 I=1,NN
        READ(5,*) NUM, COOR(NUM,1), COOR(NUM,2)
   10 CONTINUE
C
C  READ ELEMENT CONNECTIVITY
C
      DO 15 I=1,NE
        READ(5,*) NUM, (NO(NUM,J), J=1,NNODE)
   15 CONTINUE
C
C  READ BOUNDARY CONDITIONS
C
      DO 25 I=1,NBOUN
        READ(5,*) (IBORD(J,I),J=1,4)
   25 CONTINUE
C
C  VALUES AT INFLOW
C
      CINF(1)=1.D0
      CINF(2)=1.D0*COSD(ALF)
      CINF(3)=1.D0*SIND(ALF)
      CINF(4)=1.D0
```

```
      CINF(5)=1.D0
C
C  SET INITIAL CONDITIONS
C
      IF (ISTART .EQ. 0) THEN
        DO 180 I=1,NN
          RHO1(I)=CINF(1)
          U1(I)=CINF(2)
          V1(I)=CINF(3)
          E1(I)=CINF(4)
 180    CONTINUE
      ELSE
        READ(4,500)TEXT
        READ(4,*)ITOTAL
        DO 190 I=1,NN
          READ(4,*)N,RHO1(N),U1(N),V1(N),E1(N),ETOTAL,AM
 190    CONTINUE
      END IF
      DO 200 I=1,NN
        RHO2(I)=RHO1(I)
        U2(I)=U1(I)
        V2(I)=V1(I)
        E2(I)=E1(I)
 200  CONTINUE
      RETURN
C
C  FORMAT
C
 500  FORMAT(A60)
      END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE BINFO(NN,NBOUN,NB,
     1 UNEW, VNEW, RHONEW, ENEW, UOLD, VOLD, RHOOLD, EOLD)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  THIS SUBROUTINE CALCULATES BOUNDARY INFORMATION
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
      DIMENSION MFLAG(2,NDNODE), MFLAGT(2,20), UNORMS(2,NDNODE),
     1 UNORMT(2,20),
     1 UNEW(NDNODE), VNEW(NDNODE), RHONEW(NDNODE), ENEW(NDNODE),
     2 UOLD(NDNODE), VOLD(NDNODE), RHOOLD(NDNODE), EOLD(NDNODE)
C
C  INITIALIZE SOME VARIABLES
C
      DO 150 I=1,NDNODE
        DO 150 J=1,2
 150  UNORMS(J,I)=0.D0
```

```
      DO 160 I=1,2
        DO 160 J=1,20
 160  UNORMT(J,I)=0.D0
      DO 175 I=1,NDNODE
 175  MFLAG(2,I)=0
      NB=0
      NC=0
      NCT=0
C
      DO 200 I=1,NBOUN
        NODE1=IBORD(1,I)
        NODE2=IBORD(2,I)
        IFL=IBORD(4,I)
        DELX=COOR(NODE2,1)-COOR(NODE1,1)
        DELY=COOR(NODE2,2)-COOR(NODE1,2)
        D=DSQRT(DELX*DELX+DELY*DELY)
        IF (MFLAG(1,NODE1) .NE. IFL) THEN
          IF (MFLAG(1,NODE1) .EQ. 0) GO TO 240
          NCT=NCT+1
          UNORMT(1,NCT)=DELY/D
          UNORMT(2,NCT)=-DELX/D
          MFLAGT(1,NCT)=NODE1
          MFLAGT(2,NCT)=IFL
          GO TO 250
        END IF
 240    UNORMS(1,NODE1)=UNORMS(1,NODE1)+DELY/D
        UNORMS(2,NODE1)=UNORMS(2,NODE1)-DELX/D
        MFLAG(1,NODE1)=IFL
        MFLAG(2,NODE1)=MFLAG(2,NODE1)+1
 250    IF (MFLAG(1,NODE2) .NE. IFL) THEN
          IF (MFLAG(1,NODE2) .EQ. 0) GO TO 260
          NCT=NCT+1
          UNORMT(1,NCT)=DELY/D
          UNORMT(2,NCT)=-DELX/D
          MFLAGT(1,NCT)=NODE2
          MFLAGT(2,NCT)=IFL
          GO TO 200
        END IF
 260    UNORMS(1,NODE2)=UNORMS(1,NODE2)+DELY/D
        UNORMS(2,NODE2)=UNORMS(2,NODE2)-DELX/D
        MFLAG(1,NODE2)=IFL
        MFLAG(2,NODE2)=MFLAG(2,NODE2)+1
 200  CONTINUE
      DO 300 I=1,NN
        IF (MFLAG(1,I) .NE. 0) THEN
          IF (MFLAG(1,I) .EQ. 1) NEB=NEB+1
          NC=NC+1
          NENODE(1,NC)=I
          NENODE(2,NC)=MFLAG(1,I)
          UNORM(NC,1)=UNORMS(1,I)/REAL(MFLAG(2,I))
          UNORM(NC,2)=UNORMS(2,I)/REAL(MFLAG(2,I))
        END IF
 300  CONTINUE
      DO 400 I=1,NCT
        N=NC+I
        NENODE(1,N)=MFLAGT(1,I)
        NENODE(2,N)=MFLAGT(2,I)
        UNORM(N,1)=UNORMT(1,I)
        UNORM(N,2)=UNORMT(2,I)
        IF (MFLAGT(1,I) .EQ. 1) NEB=NEB+1
 400  CONTINUE
      NB=NC+NCT
      RETURN
      END
C
C
```

```
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE ELEMENT(NE)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES SHAPE FUNCTIONS, THEIR DERIVATIVES AND JACOBIANS
C  AT GAUSSIAN POINTS. ALSO CALCULATES ELEMENT LENGTH VECTORS.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      DIMENSION G(2), X(4), Y(4), A(2,2), B(3,3)
C
      GOTO (1,2,3) NELTYPE
C
C  SHAPE FUNCTIONS
C
C
C  BILINEAR QUADRILATERALS
C
 1    G(1)=-.57735026918963D0
      G(2)=-G(1)
      W(1)=1.D0
      W(2)=1.D0
      W(3)=1.D0
      W(4)=1.D0
      DO 10 I=1,2
        DO 15 J=1,2
          K=I+I+J-2
          P(1,K)=(1.-G(I))*(1.-G(J))*.25D0
          P(2,K)=(1.+G(I))*(1.-G(J))*.25D0
          P(3,K)=(1.+G(I))*(1.+G(J))*.25D0
          P(4,K)=(1.-G(I))*(1.+G(J))*.25D0
 15     CONTINUE
 10   CONTINUE
      DO 20 I=1,NE
        DO 30 J=1,4
          X(J)=COOR(NO(I,J),1)
 30       Y(J)=COOR(NO(I,J),2)
        X3MX4=X(3)-X(4)
        X3MX2=X(3)-X(2)
        X2MX1=X(2)-X(1)
        X4MX1=X(4)-X(1)
        Y3MY4=Y(3)-Y(4)
        Y3MY2=Y(3)-Y(2)
        Y2MY1=Y(2)-Y(1)
        Y4MY1=Y(4)-Y(1)
C
C  JACOBIAN
C
        DO 40 J=1,2
          DO 40 K=1,2
            M=J+J+K-2
            A(1,1)=((1.+G(K))*X3MX4+(1.-G(K))*X2MX1)*.25D0
            A(1,2)=((1.+G(K))*Y3MY4+(1.-G(K))*Y2MY1)*.25D0
            A(2,1)=((1.+G(J))*X3MX2+(1.-G(J))*X4MX1)*.25D0
            A(2,2)=((1.+G(J))*Y3MY2+(1.-G(J))*Y4MY1)*.25D0
            DA(I,M)=A(1,1)*A(2,2)-A(1,2)*A(2,1)
C
C  JACOBIAN INVERSE
```

```
C
                EJACOB(I,1,1,M)= A(2,2)/DA(I,M)
                EJACOB(I,1,2,M)=-A(1,2)/DA(I,M)
                EJACOB(I,2,1,M)=-A(2,1)/DA(I,M)
                EJACOB(I,2,2,M)= A(1,1)/DA(I,M)
C
C  SHAPE FUNCTION DERIVATIVES
C
                DP(1,M,1,I)=(-A(2,2)*(1.-G(K))+A(1,2)*(1.-G(J)))/DA(I,M)
                DP(1,M,2,I)=( A(2,1)*(1.-G(K))-A(1,1)*(1.-G(J)))/DA(I,M)
                DP(2,M,1,I)=( A(2,2)*(1.-G(K))+A(1,2)*(1.+G(J)))/DA(I,M)
                DP(2,M,2,I)=(-A(2,1)*(1.-G(K))-A(1,1)*(1.+G(J)))/DA(I,M)
                DP(3,M,1,I)=( A(2,2)*(1.+G(K))-A(1,2)*(1.+G(J)))/DA(I,M)
                DP(3,M,2,I)=(-A(2,1)*(1.+G(K))+A(1,1)*(1.+G(J)))/DA(I,M)
                DP(4,M,1,I)=(-A(2,2)*(1.+G(K))-A(1,2)*(1.-G(J)))/DA(I,M)
                DP(4,M,2,I)=( A(2,1)*(1.+G(K))+A(1,1)*(1.-G(J)))/DA(I,M)
                DA(I,M)=DA(I,M)*W(M)
  40        CONTINUE
  20      CONTINUE
          GO TO 1000
C
C  LINEAR/LINEAR TRIANGLES
C
   2      G(1)=1.D0/3.D0
          P(1,1)=1.-G(1)-G(1)
          P(2,1)=G(1)
          P(3,1)=G(1)
          W(1)=1.D0
          DO 120 I=1,NE
            DO 130 J=1,3
              X(J)=COOR(NO(I,J),1)
 130          Y(J)=COOR(NO(I,J),2)
            X2MX1=X(2)-X(1)
            Y2MY1=Y(2)-Y(1)
            X3MX1=X(3)-X(1)
            Y3MY1=Y(3)-Y(1)
C
C  JACOBIAN
C
            A(1,1)=X2MX1
            A(1,2)=Y2MY1
            A(2,1)=X3MX1
            A(2,2)=Y3MY1
            DA(I,1)=A(1,1)*A(2,2)-A(1,2)*A(2,1)
C
C  JACOBIAN INVERSE
C
            EJACOB(I,1,1,1)= A(2,2)/DA(I,1)
            EJACOB(I,1,2,1)=-A(1,2)/DA(I,1)
            EJACOB(I,2,1,1)=-A(2,1)/DA(I,1)
            EJACOB(I,2,2,1)= A(1,1)/DA(I,1)
C
C  SHAPE FUNCTION DERIVATIVES
C
            DP(1,1,1,I)=(-A(2,2)+A(1,2))/DA(I,1)
            DP(1,1,2,I)=( A(2,1)-A(1,1))/DA(I,1)
            DP(2,1,1,I)=  A(2,2)/DA(I,1)
            DP(2,1,2,I)= -A(2,1)/DA(I,1)
            DP(3,1,1,I)= -A(1,2)/DA(I,1)
            DP(3,1,2,I)=  A(1,1)/DA(I,1)
            DA(I,1)=DA(I,1)*W(1)
 120      CONTINUE
          GO TO 1000
C
C  LINEAR/LINEAR TRIANGLES (THREE POINT INTEGRATION RULE)
C
```

```
    3   W(1)=1./3.
        W(2)=1./3.
        W(3)=1./3.
        B(1,1)=.5
        B(2,1)=.5
        B(3,1)=0.
        B(1,2)=0.
        B(2,2)=.5
        B(3,2)=.5
        DO 210 I=1,3
          P(1,I)=1.-B(I,1)-B(I,2)
          P(2,I)=B(I,1)
          P(3,I)=B(I,2)
  210   CONTINUE
        DO 220 I=1,NE
          DO 230 J=1,3
            X(J)=COOR(NO(I,J),1)
  230       Y(J)=COOR(NO(I,J),2)
          X2MX1=X(2)-X(1)
          Y2MY1=Y(2)-Y(1)
          X3MX1=X(3)-X(1)
          Y3MY1=Y(3)-Y(1)
C
C  JACOBIAN
C
          A(1,1)=X2MX1
          A(1,2)=Y2MY1
          A(2,1)=X3MX1
          A(2,2)=Y3MY1
          DO 215 J=1,3
            DA(I,J)=A(1,1)*A(2,2)-A(1,2)*A(2,1)
C
C  JACOBIAN INVERSE
C
            EJACOB(I,1,1,J)= A(2,2)/DA(I,J)
            EJACOB(I,1,2,J)=-A(1,2)/DA(I,J)
            EJACOB(I,2,1,J)=-A(2,1)/DA(I,J)
            EJACOB(I,2,2,J)= A(1,1)/DA(I,J)
C
C  SHAPE FUNCTION DERIVATIVES
C
            DP(1,J,1,I)=(-A(2,2)+A(1,2))/DA(I,J)
            DP(1,J,2,I)=( A(2,1)-A(1,1))/DA(I,J)
            DP(2,J,1,I)=  A(2,2)/DA(I,J)
            DP(2,J,2,I)= -A(2,1)/DA(I,J)
            DP(3,J,1,I)= -A(1,2)/DA(I,J)
            DP(3,J,2,I)=  A(1,1)/DA(I,J)
            DA(I,J)=DA(I,J)*W(J)
  215     CONTINUE
  220   CONTINUE
 1000   RETURN
        END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        SUBROUTINE ASSEM1(NN,NE,RHONEW,UNEW,VNEW,ENEW,F)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES AND ASSEMBLES BOTH SIDES OF THE GLOBAL VECTORS FOR THE
C  NAVIER-STOKES EQUATIONS.
C
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (NDNODE=5000, NDELEM=5000)
C
```

```
            EY=EY+DP(J,NG,2,I)*ENEW(NODE)
            PY=PY+DP(J,NG,2,I)*RHONEW(NODE)*ENEW(NODE)
            ZZ=ZZ+P(J,NG)
   30     CONTINUE
C
C  PERTURBATION COEFFICIENT FOR THE CONTINUITY EQUATION
C
          U2=U*U
          V2=V*V
          UABS2=U2+V2
          HA=U*EJACOB(I,1,1,NG)+V*EJACOB(I,1,2,NG)
          HB=U*EJACOB(I,2,1,NG)+V*EJACOB(I,2,2,NG)
          H=UABS2/SQRT(HA*HA+HB*HB)*F
          COFC=H*.5D0/UABS2
C
C  PERTURBATION COEFFICIENT FOR THE X-MOMENTUM EQUATION
C
          GAMU=RHO*RE*H/(1.D0+U2/(3.D0*UABS2))
          COFU=ALPHA(GAMU)*H*.5D0/UABS2
C
C  PERTURBATION COEFFICIENT FOR THE Y-MOMENTUM EQUATION
C
          GAMV=RHO*RE*H/(1.D0+V2/(3.D0*UABS2))
          COFV=ALPHA(GAMV)*H*.5D0/UABS2
C
C  PERTURBATION COEFFICIENT FOR ENERGY EQUATION
C
          GAME=RHO*H/C2
          COFE=ALPHA(GAME)*H*.5D0/UABS2
C
C  CALCULATE CONVECTIVE TERMS
C
          RHOCON=U*RHOX+V*RHOY
          UCON=RHO*(U*UX+V*UY)
          VCON=RHO*(U*VX+V*VY)
          ECON=RHO*(U*EX+V*EY)
C
C  CALCULATE FORCING TERMS
C
          RHOFOR=RHO*(UX+VY)
          UFOR=ONEDGM2*PX
          VFOR=ONEDGM2*PY
          EFOR=GM1*E*RHO*(UX+VY)
C
C  CALCULATE VISCOUS TERMS
C
          UVIS1=(4.D0*UX-2.D0*VY)/3.D0
          UVIS2=UY+VX
          VVIS1=UVIS2
          VVIS2=(4.D0*VY-2.D0*UX)/3.D0
          EVIS=C1*(4.D0/3.D0*(UX*UX+VY*VY-UX*VY)+
     1         UY*UY+VX*VX+2.D0*UY*VX)
C
C  COMPUTE UPWIND WEIGHTING FUNCTIONS
C
          DO 60 J=1,NNODE
            NODE=NO(I,J)
            UDP=U*DP(J,NG,1,I)
            VDP=V*DP(J,NG,2,I)
            UDPPVDP=UDP+VDP
```

```
C
               RHR(NODE)=RHR(NODE)-PC*(RHOCON+RHOFOR)*DA(I,NG)
               RHU(NODE)=RHU(NODE)-(PU*(UCON+UFOR)+
     1            (DP(J,NG,1,I)*UVIS1+DP(J,NG,2,I)*UVIS2)/RE)*DA(I,NG)
               RHV(NODE)=RHV(NODE)-(PV*(VCON+VFOR)+
     1            (DP(J,NG,1,I)*VVIS1+DP(J,NG,2,I)*VVIS2)/RE)*DA(I,NG)
               RHE(NODE)=RHE(NODE)-(PE*(ECON+EFOR-EVIS)+
     2            (DP(J,NG,1,I)*EX+DP(J,NG,2,I)*EY)*C2)*DA(I,NG)
C
C  ASSEMBLE LEFT HAND SIDE (LUMPED MASSES)
C
               XLHR(NODE)=XLHR(NODE)+PC*ZZ*DA(I,NG)
               XLHU(NODE)=XLHU(NODE)+PU*ZZ*RHO*DA(I,NG)
               XLHV(NODE)=XLHV(NODE)+PV*ZZ*RHO*DA(I,NG)
               XLHE(NODE)=XLHE(NODE)+PE*ZZ*RHO*DA(I,NG)
   60        CONTINUE
   20      CONTINUE
   10    CONTINUE
         RETURN
         END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
         SUBROUTINE ASSEM2(NN,NE,RHONEW,UNEW,VNEW,ENEW,F)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES AND ASSEMBLES BOTH SIDES OF THE GLOBAL VECTORS FOR THE
C  EULER EQUATIONS.
C
         IMPLICIT REAL*8 (A-H,O-Z)
         PARAMETER (NDNODE=5000, NDELEM=5000)
C
         COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
         COMMON /SOLN/ NMASS(20,NDNODE),
     1   XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2   RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3   XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
         COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1   NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
         COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1   MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
         COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2
C
         DIMENSION RHONEW(NDNODE), UNEW(NDNODE), VNEW(NDNODE),
     1   ENEW(NDNODE)
C
C        IF (NELTYPE .EQ. 1) F=2.D0
C        IF (NELTYPE .EQ. 2) F=.5D0
         DO 5 I=1,NN
           RHR(I)=0.D0
           RHU(I)=0.D0
           RHV(I)=0.D0
           RHE(I)=0.D0
           XLHR(I)=0.D0
           XLHU(I)=0.D0
           XLHV(I)=0.D0
           XLHE(I)=0.D0
           NMASS(1,I)=1
           DO 5 J=1,20
             XMC(J,I)=0.D0
```

```
            XMU(J,I)=0.D0
            XMV(J,I)=0.D0
            XME(J,I)=0.D0
    5   CONTINUE
        DO 10 I=1,NE
          DO 20 NG=1,NGAUSS
            RHO=0.D0
            U=0.D0
            V=0.D0
            E=0.D0
            RHOX=0.D0
            UX=0.D0
            VX=0.D0
            EX=0.D0
            RHOY=0.D0
            UY=0.D0
            VY=0.D0
            EY=0.D0
            ZZ=0.D0
C
C   EVALUATE QUANTITIES AT GAUSSIAN POINTS
C
            DO 30 J=1,NNODE
              NODE=NO(I,J)
              RHO=RHO+P(J,NG)*RHONEW(NODE)
              U=U+P(J,NG)*UNEW(NODE)
              V=V+P(J,NG)*VNEW(NODE)
              E=E+P(J,NG)*ENEW(NODE)
              RHOX=RHOX+DP(J,NG,1,I)*RHONEW(NODE)
              UX=UX+DP(J,NG,1,I)*UNEW(NODE)
              VX=VX+DP(J,NG,1,I)*VNEW(NODE)
              EX=EX+DP(J,NG,1,I)*ENEW(NODE)
              RHOY=RHOY+DP(J,NG,2,I)*RHONEW(NODE)
              UY=UY+DP(J,NG,2,I)*UNEW(NODE)
              VY=VY+DP(J,NG,2,I)*VNEW(NODE)
              EY=EY+DP(J,NG,2,I)*ENEW(NODE)
              ZZ=ZZ+P(J,NG)
   30       CONTINUE
C
C   PERTURBATION COEFFICIENT
C
            U2=U*U
            V2=V*V
            UABS2=U2+V2
            UABS=DSQRT(UABS2)
            UL1=U*EJACOB(I,1,1,NG)+V*EJACOB(I,2,1,NG)
            UL2=U*EJACOB(I,1,2,NG)+V*EJACOB(I,2,2,NG)
            ULABS=DSQRT(UL1*UL1+UL2*UL2)
            IF (ULABS .LT. 1.D-10) THEN
              COF=0.D0
              GO TO 35
            END IF
            COF=(UABS/ULABS)*F/(2.D0*UABS)
C
C   CALCULATE CONVECTIVE TERMS
C
   35       RHOCON=U*RHOX+V*RHOY
            UCON=RHO*(U*UX+V*UY)
            VCON=RHO*(U*VX+V*VY)
            ECON=RHO*(U*EX+V*EY)
C
C   CALCULATE FORCING TERMS
C
            PX=RHO*EX+RHOX*E
            PY=RHO*EY+RHOY*E
            RHOFOR=RHO*(UX+VY)
```

```
          UFOR=ONEDGM2*PX
          VFOR=ONEDGM2*PY
          EFOR=GM1*E*RHO*(UX+VY)
C
C  COMPUTE DISCONTINUITY CAPTURING TERMS IF NEEDED
C
          IF (MUPW .LT. 3) GO TO 39
          RHOX2=RHOX*RHOX
          RHOY2=RHOY*RHOY
          UX2=UX*UX
          UY2=UY*UY
          VX2=VX*VX
          VY2=VY*VY
          EX2=EX*EX
          EY2=EY*EY
          DRHO2=RHOX2+RHOY2
          DU2=UX2+UY2
          DV2=VX2+VY2
          DE2=EX2+EY2
          DRHO=DSQRT(DRHO2)
          DU=DSQRT(DU2)
          DV=DSQRT(DV2)
          DE=DSQRT(DE2)
C
C  CONTINUITY
C
          IF (DRHO2 .LT. 1.D-5) GO TO 36
          UGR=RHOCON*RHOX/DRHO2
          VGR=RHOCON*RHOY/DRHO2
          UG2=UGR*UGR
          VG2=VGR*VGR
          UABSG2=UG2+VG2
          UABSG=DSQRT(UABSG2)
          UL1=UGR*EJACOB(I,1,1,NG)+VGR*EJACOB(I,2,1,NG)
          UL2=UGR*EJACOB(I,1,2,NG)+VGR*EJACOB(I,2,2,NG)
          ULABS=DSQRT(UL1*UL1+UL2*UL2)
          IF (ULABS .LT. 1.D-10) THEN
            COF1=0.D0
            GO TO 36
          END IF
          HG=UABSG/ULABS*F
          COF1=HG*.5D0/UABSG
C
C  X-MOMENTUM
C
   36     IF (DU2 .LT. 1.D-5) GO TO 37
          UGU=UCON/RHO*UX/DU2
          VGU=UCON/RHO*UY/DU2
          UG2=UGU*UGU
          VG2=VGU*VGU
          UABSG2=UG2+VG2
          UABSG=DSQRT(UABSG2)
          UL1=UGU*EJACOB(I,1,1,NG)+VGU*EJACOB(I,2,1,NG)
          UL2=UGU*EJACOB(I,1,2,NG)+VGU*EJACOB(I,2,2,NG)
          ULABS=DSQRT(UL1*UL1+UL2*UL2)
          IF (ULABS .LT. 1.D-10) THEN
            COF2=0.D0
            GO TO 37
          END IF
          HG=UABSG/ULABS*F
          COF2=HG*.5D0/UABSG
C
C  Y-MOMENTUM
C
   37     IF (DV2 .LT. 1.D-5) GO TO 38
          UGV=VCON/RHO*VX/DV2
```

```
            VGV=VCON/RHO*VY/DV2
            UG2=UGV*UGV
            VG2=VGV*VGV
            UABSG2=UG2+VG2
            UABSG=DSQRT(UABSG2)
            UL1=UGV*EJACOB(I,1,1,NG)+VGV*EJACOB(I,2,1,NG)
            UL2=UGV*EJACOB(I,1,2,NG)+VGV*EJACOB(I,2,2,NG)
            ULABS=DSQRT(UL1*UL1+UL2*UL2)
            IF (ULABS .LT. 1.D-10) THEN
              COF3=0.D0
              GO TO 38
            END IF
            HG=UABSG/ULABS*F
            COF3=HG*.5D0/UABSG
C
C   ENERGY
C
   38       IF (DE2 .LT. 1.D-5) GO TO 39
            UGE=ECON/RHO*EX/DE2
            VGE=ECON/RHO*EY/DE2
            UG2=UGE*UGE
            VG2=VGE*VGE
            UABSG2=UG2+VG2
            UABSG=DSQRT(UABSG2)
            UL1=UGE*EJACOB(I,1,1,NG)+VGE*EJACOB(I,2,1,NG)
            UL2=UGE*EJACOB(I,1,2,NG)+VGE*EJACOB(I,2,2,NG)
            ULABS=DSQRT(UL1*UL1+UL2*UL2)
            IF (ULABS .LT. 1.D-10) THEN
              COF4=0.D0
              GO TO 39
            END IF
            HG=UABSG/ULABS*F
            COF4=HG*.5D0/UABSG
C
C   COMPUTE UPWIND WEIGHTING FUNCTION
C
   39       DO 60 J=1,NNODE
              NODE1=NO(I,J)
              UDP=U*DP(J,NG,1,I)
              VDP=V*DP(J,NG,2,I)
              UDPPVDP=UDP+VDP
              PC=P(J,NG)+COF*UDPPVDP
              IF (MUPW .EQ. 3) THEN
                UDP=UGR*DP(J,NG,1,I)
                VDP=VGR*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC1=PC+COF1*UDPPVDP
                UDP=UGU*DP(J,NG,1,I)
                VDP=VGU*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC2=PC+COF2*UDPPVDP
                UDP=UGV*DP(J,NG,1,I)
                VDP=VGV*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC3=PC+COF3*UDPPVDP
                UDP=UGE*DP(J,NG,1,I)
                VDP=VGE*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC4=PC+COF4*UDPPVDP
              END IF
C
C   ASSEMBLE RIGHT HAND SIDE
C
              IF (MUPW .EQ. 1) THEN
                RHR(NODE1)=RHR(NODE1)-(PC*RHOCON+P(J,NG)*RHOFOR)*DA(I,NG)
                RHU(NODE1)=RHU(NODE1)-(PC*UCON+P(J,NG)*UFOR)*DA(I,NG)
```

```
                RHV(NODE1)=RHV(NODE1)-(PC*VCON+P(J,NG)*VFOR)*DA(I,NG)
                RHE(NODE1)=RHE(NODE1)-(PC*ECON+P(J,NG)*EFOR)*DA(I,NG)
              ELSEIF (MUPW .EQ. 2) THEN
                RHR(NODE1)=RHR(NODE1)-PC*(RHOCON+RHOFOR)*DA(I,NG)
                RHU(NODE1)=RHU(NODE1)-PC*(UCON+UFOR)*DA(I,NG)
                RHV(NODE1)=RHV(NODE1)-PC*(VCON+VFOR)*DA(I,NG)
                RHE(NODE1)=RHE(NODE1)-PC*(ECON+EFOR)*DA(I,NG)
              ELSE
                RHR(NODE1)=RHR(NODE1)-PC1*(RHOCON+RHOFOR)*DA(I,NG)
                RHU(NODE1)=RHU(NODE1)-PC2*(UCON+UFOR)*DA(I,NG)
                RHV(NODE1)=RHV(NODE1)-PC3*(VCON+VFOR)*DA(I,NG)
                RHE(NODE1)=RHE(NODE1)-PC4*(ECON+EFOR)*DA(I,NG)
              END IF
C
C  ASSEMBLE LEFT HAND SIDE (CONSISTENT MASSES)
C
              IF (NCON .GT. 1) THEN
                NPASS=0
                DO 70 K=1,NNODE
                  NODE2=NO(I,K)
                  IF (MUPW .EQ. 1) THEN
                    TC=P(J,NG)*P(K,NG)*DA(I,NG)
                    TU=P(J,NG)*P(K,NG)*RHO*DA(I,NG)
                    TV=TU
                    TE=TU
                  ELSEIF (MUPW .EQ. 2) THEN
                    TC=PC*P(K,NG)*DA(I,NG)
                    TU=PC*P(K,NG)*RHO*DA(I,NG)
                    TV=TU
                    TE=TU
                  ELSE
                    TC=PC1*P(K,NG)*DA(I,NG)
                    TU=PC2*P(K,NG)*RHO*DA(I,NG)
                    TV=PC3*P(K,NG)*RHO*DA(I,NG)
                    TE=PC4*P(K,NG)*RHO*DA(I,NG)
                  END IF
                  IF (NODE1 .EQ. NODE2) THEN
                    XMC(1,NODE1)=XMC(1,NODE1)+TC
                    XMU(1,NODE1)=XMU(1,NODE1)+TU
                    XMV(1,NODE1)=XMV(1,NODE1)+TV
                    XME(1,NODE1)=XME(1,NODE1)+TE
                    GO TO 70
                  END IF
                  NPASS=NPASS+1
                  N1=NMASS(1,NODE1)+NPASS
                  XMC(N1,NODE1)=XMC(N1,NODE1)+TC
                  XMU(N1,NODE1)=XMU(N1,NODE1)+TU
                  XMV(N1,NODE1)=XMV(N1,NODE1)+TV
                  XME(N1,NODE1)=XME(N1,NODE1)+TE
                  NMASS(N1,NODE1)=NODE2
   70           CONTINUE
              END IF
C
C  LUMPED MASSES
C
            IF (MUPW .EQ. 1) THEN
              XLHR(NODE1)=XLHR(NODE1)+P(J,NG)*ZZ*DA(I,NG)
              XLHU(NODE1)=XLHU(NODE1)+P(J,NG)*ZZ*RHO*DA(I,NG)
              XLHV(NODE1)=XLHU(NODE1)
              XLHE(NODE1)=XLHU(NODE1)
            ELSEIF (MUPW .EQ. 2) THEN
              XLHR(NODE1)=XLHR(NODE1)+PC*ZZ*DA(I,NG)
              XLHU(NODE1)=XLHU(NODE1)+PC*ZZ*RHO*DA(I,NG)
              XLHV(NODE1)=XLHU(NODE1)
              XLHE(NODE1)=XLHU(NODE1)
            ELSE
```

```
            XLHR(NODE1)=XLHR(NODE1)+PC1*ZZ*DA(I,NG)
            XLHU(NODE1)=XLHU(NODE1)+PC2*ZZ*RHO*DA(I,NG)
            XLHV(NODE1)=XLHV(NODE1)+PC3*ZZ*RHO*DA(I,NG)
            XLHE(NODE1)=XLHE(NODE1)+PC4*ZZ*RHO*DA(I,NG)
          END IF
   60     CONTINUE
   20   CONTINUE
        IF (NCON .GT. 1) THEN
          DO 80 J=1,NNODE
            NODE=NO(I,J)
            NMASS(1,NODE)=NMASS(1,NODE)+(NNODE-1)
   80     CONTINUE
        END IF
   10 CONTINUE
      RETURN
      END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE SOLVE(NN,NB,
     1                 RHO1,U1,V1,E1,
     2                 RHO2,U2,V2,E2,TIME)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  COMPUTES VALUES OF RHO, U, V AND E AT NEXT TIME STEP
C  AND INCORPORATES THE BOUNDARY CONDITIONS.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2
C
      COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
      DIMENSION RHO1(NDNODE), U1(NDNODE), V1(NDNODE), E1(NDNODE),
     1 RHO2(NDNODE), U2(NDNODE), V2(NDNODE), E2(NDNODE), TIME(NDNODE),
     2 DEL0(4,NDNODE), DEL1(4,NDNODE), DEL2(4,NDNODE)
C
C  INCREMENT OF DEPENDENT VARIABLES USING LUMPED MASS
C
      DO 20 I=1,NN
        DEL0(1,I)=TIME(I)*RHR(I)/XLHR(I)
        DEL0(2,I)=TIME(I)*RHU(I)/XLHU(I)
        DEL0(3,I)=TIME(I)*RHV(I)/XLHV(I)
        DEL0(4,I)=TIME(I)*RHE(I)/XLHE(I)
   20 CONTINUE
C
C  LUMPED MASS SOLUTION
C
      IF (NCON .EQ. 1) THEN
        DO 30 I=1,NN
          RHO2(I)=RHO1(I)+DEL0(1,I)
          U2(I)=U1(I)+DEL0(2,I)
          V2(I)=V1(I)+DEL0(3,I)
```

```
          E2(I)=E1(I)+DEL0(4,I)
  30    CONTINUE
        GO TO 100
      END IF
C
C  ITERATIVE SCHEME FOR USE OF CONSISTENT MASS MATRIX
C
      DO 45 I=1,4
        DO 45 J=1,NN
  45  DEL1(I,J)=DEL0(I,J)
      DO 50 ICON=2,NCON
        DO 70 I=1,NN
          DEL2(1,I)=DEL0(1,I)+(1.D0-XMC(1,I)/XLHR(I))*DEL1(1,I)
          DEL2(2,I)=DEL0(2,I)+(1.D0-XMU(1,I)/XLHU(I))*DEL1(2,I)
          DEL2(3,I)=DEL0(3,I)+(1.D0-XMV(1,I)/XLHV(I))*DEL1(3,I)
          DEL2(4,I)=DEL0(4,I)+(1.D0-XME(1,I)/XLHE(I))*DEL1(4,I)
          NUM=NMASS(1,I)
          DO 80 J=2,NUM
            NN=NMASS(J,I)
            DEL2(1,NN)=DEL0(1,NN)-XMC(J,NN)*DEL1(1,NN)
            DEL2(2,NN)=DEL0(2,NN)-XMU(J,NN)*DEL1(2,NN)
            DEL2(3,NN)=DEL0(3,NN)-XMV(J,NN)*DEL1(3,NN)
            DEL2(4,NN)=DEL0(4,NN)-XME(J,NN)*DEL1(4,NN)
  80      CONTINUE
  70    CONTINUE
        IF (ICON .LT. NCON) THEN
          DO 85 I=1,4
            DO 85 J=1,NN
  85      DEL1(I,J)=DEL2(I,J)
        END IF
  50  CONTINUE
      DO 90 I=1,NN
        RHO2(I)=RHO1(I)+DEL2(1,I)
        U2(I)=U1(I)+DEL2(2,I)
        V2(I)=V1(I)+DEL2(3,I)
        E2(I)=E1(I)+DEL2(4,I)
  90  CONTINUE
C
C  ENFORCE BOUNDARY CONDITIONS
C
 100  TW=1.D0
      RI=CINF(1)
      UI=CINF(2)
      VI=CINF(3)
      EI=CINF(4)
      PI=CINF(5)
      DO 40 I=1,NB
        NUM=NENODE(1,I)
        RN=RHO1(NUM)
        UN=U1(NUM)
        VN=V1(NUM)
        EN=E1(NUM)
        PN=RN*EN
        RP=RHO2(NUM)
        UP=U2(NUM)
        VP=V2(NUM)
        EP=E2(NUM)
        PP=RP*EP
        XN=UNORM(I,1)
        YN=UNORM(I,2)
        IFL=NENODE(2,I)
        GOTO (300,400,500,600) IFL
C
C  "FREE" BOUNDARY
C
 300    UN2=UN*UN
```

```
      VN2=VN*VN
      UNABS2=UN2+VN2
      SN2=EN/(XMACH*XMACH)
      XM2=UNABS2/SN2
      SN=DSQRT(SN2)
C
C  NORMAL AND TANGENTIAL BOUNDARY VELOCITIES
C
      UNP=UP*XN+VP*YN
      UTP=-UP*YN+VP*XN
      UNN=UN*XN+VN*YN
      UTN=-UN*YN+VN*XN
      UNI=UI*XN+VI*YN
      UTI=-UI*YN+VI*XN
C
C  OUTFLOW
C
      IF (UNP .GE. 0.D0) THEN
C
C  SUPERSONIC
C
        IF (XM2 .GT. 1.D0) GO TO 40
C
C  SUBSONIC
C
        DELP=PI-PP
        RHO2(NUM)=RP+DELP*ONEDGM2/SN2
        UNC=UNP+DELP*ONEDGM2/(RN*SN)
        UTC=UTP
        U2(NUM)=UNC*XN-UTC*YN
        V2(NUM)=UNC*YN+UTC*XN
        E2(NUM)=PI/RHO2(NUM)
        GO TO 40
      END IF
C
C  INFLOW
C
      IF (ITYPE .EQ. 0 .OR. XM2 .GT. 1.D0) THEN
C
C  NAVIER-STOKES OR SUPERSONIC EULER
C
        RHO2(NUM)=RI
        U2(NUM)=UI
        V2(NUM)=VI
        E2(NUM)=EI
        GO TO 40
      END IF
C
C  SUBSONIC EULER
C
      PC=0.5D0*(PI+PP+RN*SN*(-UNI+UNP)/ONEDGM2)
      RC=RI+(PC-PI)*ONEDGM2/SN2
      UNC=UNI-(PI-PC)*ONEDGM2/(RN*SN)
      UTC=UTI
      RHO2(NUM)=RC
      U2(NUM)=UNC*XN-UTC*YN
      V2(NUM)=UNC*YN+UTC*XN
      E2(NUM)=PC/RC
      GO TO 40
C
C  ADIABATIC WALL
C
 400  IF (ITYPE .EQ. 1) THEN
C
C  EULER
C
```

```
            U2(NUM)=UP*(1.0D0-XN*XN)-VP*XN*YN
            V2(NUM)=VP*(1.0D0-YN*YN)-UP*XN*YN
            GO TO 40
          END IF
C
C  NAVIER-STOKES
C
          U2(NUM)=0.D0
          V2(NUM)=0.D0
          GO TO 40
C
C  CONSTANT TEMPERATURE WALL
C
 500      IF (ITYPE .EQ. 1) THEN
C
C  EULER
C
            U2(NUM)=UP*(1.0D0-XN*XN)-VP*XN*YN
            V2(NUM)=VP*(1.0D0-YN*YN)-UP*XN*YN
            E2(NUM)=TW
            GO TO 40
          END IF
C
C  NAVIER-STOKES
C
          U2(NUM)=0.D0
          V2(NUM)=0.D0
          E2(NUM)=TW
          GO TO 40
C
C  SYMMETRY BOUNDARY
C
 600      U2(NUM)=UP*(1.0D0-XN*XN)-VP*XN*YN
          V2(NUM)=VP*(1.0D0-YN*YN)-UP*XN*YN
  40    CONTINUE
        RETURN
        END
C
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        SUBROUTINE STEADY(NN,ITOTAL,NEXT,CONTOT)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CHECK FOR STEADY STATE AND REASSIGN AND ASSIGN VALUE OF 1 TO
C  VARIABLE NEXT IF CONVERGENCE CRITERION IS MET.
C
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (NDNODE=5000, NDELEM=5000)
C
        COMMON /SOLN/ NMASS(20,NDNODE),
     1  XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2  RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3  XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
        DIMENSION CON(4)
C
        EPS=1.D-4
        DO 10 I=1,4
          CON(I)=0.D0
  10    CONTINUE
        DO 20 I=1,NN
          RR=RHR(I)*RHR(I)
          UU=RHU(I)*RHU(I)
          VV=RHV(I)*RHV(I)
```

```
        EE=RHE(I)*RHE(I)
        CON(1)=CON(1)+RR
        CON(2)=CON(2)+UU
        CON(3)=CON(3)+VV
        CON(4)=CON(4)+EE
 20   CONTINUE
      TOTAL=CON(1)+CON(2)+CON(3)+CON(4)
      DEL=(TOTAL-CONTOT)/TOTAL
      WRITE(6,100)ITOTAL,(CON(I),I=1,4),TOTAL
      IF (DABS(DEL) .LT. EPS) THEN
        NEXT=1
        RETURN
      END IF
      CONTOT=TOTAL
      RETURN
C
C  FORMAT
C
100   FORMAT (I5,5(1X,E9.4))
      END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE TSTEP1(NN,NE,UNEW,VNEW,ENEW,DELTP,F)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES TIME STEP FOR NAVIER-STOKES SOLVER.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2
C
      DIMENSION UNEW(NDNODE), VNEW(NDNODE), ENEW(NDNODE),
     1 DELTP(NDNODE), DELTE(NDELEM)
C
      D=REAL(NNODE)
      DO 10 I=1,NE
        U=0.D0
        V=0.D0
        E=0.D0
        DO 20 J=1,NNODE
          NODE=NO(I,J)
          U=U+UNEW(NODE)
          V=V+VNEW(NODE)
          E=E+ENEW(NODE)
 20     CONTINUE
        U=U/D
        V=V/D
        E=E/D
        U2=U*U
```

```
          V2=V*V
          UABS2=U2+V2
          HA1=U*EJACOB(I,1,1,1)+V*EJACOB(I,2,1,1)
          HB1=U*EJACOB(I,1,2,1)+V*EJACOB(I,2,2,1)
          H1=UABS2/SQRT(HA1*HA1+HB1*HB1)*F
          HA2=V*EJACOB(I,1,1,1)+U*EJACOB(I,2,1,1)
          HB2=V*EJACOB(I,1,2,1)+U*EJACOB(I,2,2,1)
          H2=UABS2/SQRT(HA2*HA2+HB2*HB2)*F
          TERM1=GAMMA*UABS2
          TERM2=SQRT(E*UABS2)/XMACH
          TERM3=2.D0*C2*UABS2/H1
          TERM4=2.D0*C2*UABS2*H1/(H2*H2)
          DELTE(I)=SAFE*H1/(TERM1+TERM2+TERM3+TERM4)
   10   CONTINUE
        NPASS=0
 3000   NPASS=NPASS+1
        DO 100 I=1,NN
  100   DELTP(I)=1.D6
        DO 5000 IE=1,NE
          DO 4000 IN=1,NNODE
            IP=NO(IE,IN)
            DELTP(IP)=MIN(DELTP(IP),DELTE(IN))
 4000     CONTINUE
 5000   CONTINUE
C
        DO 7000 IE=1,NE
        KOUNT=0
        DO 6000 IN=1,NNODE
          IP=NO(IE,IN)
          TOLER=0.8D0*DELTE(IE)-DELTP(IP)
          IF (TOLER .GT. 1.D-4) THEN
            KOUNT=KOUNT+1
            DELTE(IE)=DELTP(IP)/0.8D0
          END IF
 6000     CONTINUE
 7000   CONTINUE
C
        IF (NPASS .GE. 10) GO TO 1000
        IF (KOUNT .NE. 0) GO TO 3000
 1000   RETURN
        END
C
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        SUBROUTINE TSTEP2(NN,NE,UNEW,VNEW,ENEW,DELTP,F)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES TIME STEP FOR EULER SOLVER.
C
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (NDNODE=5000, NDELEM=5000)
C
        COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
        COMMON /SOLN/ NMASS(20,NDNODE),
     1  XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2  RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3  XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
        COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1  NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
        COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1  MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
```

```
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2
C
      DIMENSION UNEW(NDNODE), VNEW(NDNODE), ENEW(NDNODE),
     1 DELTP(NDNODE), DELTE(NDELEM)
C
      D=REAL(NNODE)
      DO 10 I=1,NE
        U=0.D0
        V=0.D0
        E=0.D0
        DO 20 J=1,NNODE
          NODE=NO(I,J)
          U=U+UNEW(NODE)
          V=V+VNEW(NODE)
          E=E+ENEW(NODE)
   20   CONTINUE
        U=U/D
        V=V/D
        E=E/D
        U2=U*U
        V2=V*V
        UABS2=U2+V2
        HA=U*EJACOB(I,1,1,1)+V*EJACOB(I,2,1,1)
        HB=U*EJACOB(I,1,2,1)+V*EJACOB(I,2,2,1)
        H=UABS2/SQRT(HA*HA+HB*HB)*F
        TERM1=GAMMA*UABS2
        TERM2=SQRT(E*UABS2)/XMACH
        DELTE(I)=SAFE*H/(TERM1+TERM2)
   10 CONTINUE
      NPASS=0
 3000 NPASS=NPASS+1
      DO 100 I=1,NN
  100 DELTP(I)=1.D6
      DO 5000 IE=1,NE
        DO 4000 IN=1,NNODE
          IP=NO(IE,IN)
          DELTP(IP)=MIN(DELTP(IP),DELTE(IN))
 4000   CONTINUE
 5000 CONTINUE
      IF (NTST .EQ. 1) GO TO 4050
      T=1.D4
      DO 5050 I=1,NN
 5050 T=MIN(T,DELTP(I))
      DO 5060 I=1,NN
 5060 DELTP(I)=T
      GO TO 1000
C
C  LOCAL TIMESTEPS
C
 4050 DO 7000 IE=1,NE
      KOUNT=0
      DO 6000 IN=1,NNODE
        IP=NO(IE,IN)
        TOLER=0.8D0*DELTE(IE)-DELTP(IP)
        IF (TOLER .GT. 1.D-4) THEN
          KOUNT=KOUNT+1
          DELTE(IE)=DELTP(IP)/0.8D0
        END IF
 6000   CONTINUE
 7000 CONTINUE
C
      IF (NPASS .GE. 10) THEN
        WRITE(6,*)' NPASS > 10 IN TSTEP2'
        STOP
      END IF
```

```
      IF (KOUNT .NE. 0) GO TO 3000
 1000 RETURN
      END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE DOUTPT(NN,TITLE,ITOTAL,RHO,U,V,E)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  PRINTS DEPENDENT VARIABLES TO RESTART FILE
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2
C
      COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
      DIMENSION U(NDNODE), V(NDNODE), RHO(NDNODE), E(NDNODE)
C
C     CHARACTER TITLE*60
C
C  REWIND UNIT 3
C
      REWIND 3
C
C  WRITE TITLE AND NUMBER OF TIME STEPS
C
      WRITE(3,500)TITLE
      WRITE(3,*)ITOTAL
C
C  OUTPUT QUANTITIES
C
      CONST=GM1/(2.D0*ONEDGM2)
      DO 100 I=1,NN
        U2=U(I)*U(I)
        V2=V(I)*V(I)
        ETOTAL=E(I)+CONST*(U2+V2)
        XM=XMACH*SQRT((U2+V2)/E(I))
        WRITE(3,560) I, RHO(I), U(I), V(I), E(I), ETOTAL, XM
  100 CONTINUE
C
C  FORMATS
C
  500 FORMAT (A60)
  560 FORMAT (I5,6(1X,F9.5))
      END
```

APPENDIX II

PROGRAM PET_T.F.

```
      PROGRAM PET_T
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  SOLUTION OF TWO-DIMENSIONAL COMPRESSIBLE FLOW BY A PETROV-
C  GALERKIN FINITE ELEMENT METHOD.
C
C
C  FIRST OR SECOND ORDER TIME INTEGRATION
C
C  ELEMENTS: (1) BILINEAR QUADRILATERALS
C            (2) LINEAR/LINEAR TRIANGLES WITH ONE POINT INTEGRATION
C            (3) LINEAR/LINEAR TRIANGLES WITH THREE POINT INTEGRATION
C
C  PETROV-GALERKIN WEIGHTING OF ALL OR ONLY CONVECTIVE TERMS
C
C  TOTAL ENERGY FORMULATION
C
C  LOCAL OR GLOBAL TIMESTEPPING
C
C  LUMPED OR CONSISTENT MASS MATRICES
C
C  WITH OR WITHOUT DISCONTINUITY CAPTURING
C
C  INFLOW/OUTFLOW EULER BOUNDARY CONDITIONS PER USAB AND MURMAN
C
C
C                    FRANK P. BRUECKNER
C                    UNIVERSITY OF ARIZONA
C                    FEBRUARY 6, 1990
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
C  DEFINE COMMON BLOCKS
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
      COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
      CHARACTER*60 INFILE, OUTFILE, TITLE
C
      DIMENSION DELT(NDNODE), TIME(NDNODE),
     1 UNEW(NDNODE), VNEW(NDNODE), RHONEW(NDNODE), ENEW(NDNODE),
     2 UHALF(NDNODE), VHALF(NDNODE), RHOHALF(NDNODE), EHALF(NDNODE),
     3 UOLD(NDNODE), VOLD(NDNODE), RHOOLD(NDNODE), EOLD(NDNODE)
C
      DATA NEXT /0/
C
C  INITIALIZE SOME VARIABLES
C
```

```
      IST=10
      ITI=10
      ISC=100
      ITOTAL=1
      ISTEADY=1
      ITIME=1
      ISCRAT=1
      TTOTAL=0.
      CONTOT=1.D3
C
C  READ CONTROL DATA
C
      READ(5,1) TITLE
      READ(5,*) NELTYPE
      READ(5,*) NN, NE, NBOUN
      WRITE(6,1) TITLE
      WRITE(6,*) ' '
C
C  SET ELEMENT PARAMETERS
C
      IF (NELTYPE .EQ. 1) THEN
        NGAUSS=4
        NNODE=4
        F=2.D0
      END IF
      IF (NELTYPE .EQ. 2) THEN
        NGAUSS=1
        NNODE=3
        F=.5D0
      END IF
      IF (NELTYPE .EQ. 3) THEN
        NGAUSS=3
        NNODE=3
        F=.5D0
      END IF
C
C  READ GLOBAL DATA AND SET INITIAL CONDITIONS
C
      CALL DINPT(NN,NE,NBOUN,RHONEW,UNEW,VNEW,ENEW,
     1            RHOOLD,UOLD,VOLD,EOLD,ITOTAL)
C
C  COMPUTE BOUNDARY INFORMATION
C
      CALL BINFO(NN,NBOUN,NB,
     1 UNEW, VNEW, RHONEW, ENEW, UOLD, VOLD, RHOOLD, EOLD)
C
C  OUTPUT INITITAL CONDITIONS TO RESTART FILE
C
      CALL DOUTPT(NN,TITLE,ITOTAL,RHONEW,UNEW,VNEW,ENEW)
C
C  CALCULATE SHAPE FUNCTION AND ELEMENT DATA
C
      CALL ELEMENT(NE)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C                      NAVIER-STOKES SOLVER
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      IF (ITYPE .EQ. 1) GO TO 140
C
C  CALCULATE TIME STEP
C
      CALL TSTEP1(NN,NE,UNEW,VNEW,ENEW,TIME,F)
   40 IF (ITIME .EQ. ITI) THEN
```

```
          CALL TSTEP1(NN,NE,UNEW,VNEW,ENEW,TIME,F)
          ITIME=0
        END IF
C
C  CALCULATE AND ASSEMBLE GLOBAL EQUATIONS
C
        CALL ASSEM1(NN,NE,RHOOLD,UOLD,VOLD,EOLD,F)
        IF (NFLOW .EQ. 1) GO TO 41
C
C  ADVANCE TO TIME T + .5 * DELTA T FOR RUNGE-KUTTA INTEGRATION
C
        DO 42 I=1,NN
  42      DELT(I)=TIME(I)*.5D0
        CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1              RHOHALF,UHALF,VHALF,EHALF,DELT)
        CALL ASSEM1(NN,NE,RHOHALF,UHALF,VHALF,EHALF,F)
C
C  ADVANCE INT TIME TO T + DELTA T
C
  41    CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1              RHONEW,UNEW,VNEW,ENEW,TIME)
        TTOTAL=TTOTAL+TIME(1)
C
C  OUTPUT RESULTS TO RESTART FILE
C
        IF (ISCRAT .EQ. ISC) THEN
          REWIND 8
          CALL DOUTPT(NN,TITLE,ITOTAL,RHONEW,UNEW,VNEW,ENEW)
          ISCRAT=0
        END IF
C
C  CHECK FOR STEADY STATE
C
        IF (ISTEADY .EQ. IST) THEN
          CALL STEADY(NN,ITOTAL,NEXT,CONTOT)
          ISTEADY=0
          IF (NEXT .EQ. 1) GO TO 1000
        END IF
C
C  REASSIGN VARIABLES
C
        DO 45 I=1,NN
          RHOOLD(I)=RHONEW(I)
          UOLD(I)=UNEW(I)
          VOLD(I)=VNEW(I)
          EOLD(I)=ENEW(I)
  45    CONTINUE
        ITOTAL=ITOTAL+1
        ISTEADY=ISTEADY+1
        ITIME=ITIME+1
        ISCRAT=ISCRAT+1
        IF (ITOTAL .GT. 10000) STOP
        GO TO 40
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C                         EULER SOLVER
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C
C  CALCULATE TIME STEP
C
 140    CALL TSTEP2(NN,NE,UNEW,VNEW,ENEW,TIME,F)
 150    IF (ITIME .EQ. ITI) THEN
          CALL TSTEP2(NN,NE,UNEW,VNEW,ENEW,TIME,F)
```

```
          ITIME=0
        END IF
C
C  CALCULATE AND ASSEMBLE GLOBAL EQUATIONS
C
        CALL ASSEM2(NN,NE,RHOOLD,UOLD,VOLD,EOLD,F)
        IF (NFLOW .EQ. 1) GO TO 141
        DO 142 I=1,NN
  142     DELT(I)=TIME(I)*.5D0
C
C  ADVANCE TO TIME T + .5 * DELTA T FOR RUNGE-KUTTA INTEGRATION
C
        CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1             RHOHALF,UHALF,VHALF,EHALF,DELT)
        CALL ASSEM2(NN,NE,RHOHALF,UHALF,VHALF,EHALF,F)
C
C  ADVANCE TO TIME T + DELTA T
C
  141   CALL SOLVE(NN,NB,RHOOLD,UOLD,VOLD,EOLD,
     1             RHONEW,UNEW,VNEW,ENEW,TIME)
        TTOTAL=TTOTAL+TIME(1)
C
C  OUTPUT RESULTS TO RESTART FILE
C
        IF (ISCRAT .EQ. ISC) THEN
          REWIND 8
          CALL DOUTPT(NN,TITLE,ITOTAL,RHONEW,UNEW,VNEW,ENEW)
          ISCRAT=0
        END IF
C
C  CHECK FOR STEADY STATE
C
        IF (ISTEADY .EQ. IST) THEN
          CALL STEADY(NN,ITOTAL,NEXT,CONTOT)
          ISTEADY=0
          IF (NEXT .EQ. 1) GO TO 1000
        END IF
C
C  REASSIGN VARIABLES
C
        DO 145 I=1,NN
          RHOOLD(I)=RHONEW(I)
          UOLD(I)=UNEW(I)
          VOLD(I)=VNEW(I)
          EOLD(I)=ENEW(I)
  145   CONTINUE
        ITOTAL=ITOTAL+1
        ISTEADY=ISTEADY+1
        ISCRAT=ISCRAT+1
        ITIME=ITIME+1
        IF (ITOTAL .GT. 8000) STOP
        GO TO 150
C
C  OUTPUT FINAL RESULTS
C
 1000   REWIND 8
        CALL DOUTPT(NN,TITLE,ITOTAL,RHONEW,UNEW,VNEW,ENEW)
        STOP
C
C  FORMATS
C
    1   FORMAT (A60)
        END
C
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
C
      SUBROUTINE DINPT(NN,NE,NBOUN,RHO1,U1,V1,E1,
     1                 RHO2,U2,V2,E2,ITOTAL)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C THIS SUBROUTINE READS IN THE GLOBAL DATA AND SETS
C DEPENDENT VARIABLES TO INITIAL CONDITIONS
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
      DIMENSION RHO1(NDNODE), U1(NDNODE), V1(NDNODE), E1(NDNODE),
     1 RHO2(NDNODE), U2(NDNODE), V2(NDNODE), E2(NDNODE)
C
      CHARACTER*60 TEXT
C
C
C READ PROGRAM OPTIONS
C
      READ (5,*) MUPW, NFLOW, NCON, NTST, ITYPE, ISTART
C
C READ FREE STREAM PARAMETERS
C
      READ (5,*) XMACH, RE, PR, GAMMA, ALF
C
C READ TIME STEP SAFETY FACTOR
C
      READ (5,*) SAFE
C
C READ NODAL COORDINATES
C
      DO 10 I=1,NN
        READ(5,*) NUM, COOR(NUM,1), COOR(NUM,2)
   10 CONTINUE
C
C READ ELEMENT CONNECTIVITY
C
      DO 15 I=1,NE
        READ(5,*) NUM, (NO(NUM,J), J=1,NNODE)
   15 CONTINUE
C
C READ BOUNDARY CONDITIONS
C
      DO 25 I=1,NBOUN
        READ(5,*) (IBORD(J,I),J=1,4)
   25 CONTINUE
C
C CALCULATE SOME CONSTANTS
C
      XM2=XMACH*XMACH
      GM1=GAMMA-1.
      TWOMG=2.-GAMMA
```

```
      ONEDGM2=1./(GAMMA*XM2)
      C1=GAMMA*GM1*GM1*XM2
      C2=GAMMA*GM1*XM2/RE
      C3=GAMMA/(PR*RE)
      C4=C1/(GM1*2.)
      C5=C3/ONEDGM2
C
C  VALUES AT INFLOW
C
      CINF(1)=1.D0
      CINF(2)=1.D0*COSD(ALF)
      CINF(3)=1.D0*SIND(ALF)
      CINF(4)=1.D0+GM1/(2.D0*ONEDGM2)*(CINF(2)**2+CINF(3)**2)
      CINF(5)=1.D0
C
C  SET INITIAL CONDITIONS
C
      IF (ISTART .EQ. 0) THEN
        DO 180 I=1,NN
          RHO1(I)=CINF(1)
          U1(I)=CINF(2)
          V1(I)=CINF(3)
          E1(I)=CINF(4)
 180    CONTINUE
      ELSE
        READ(4,500)TEXT
        READ(4,*)ITOTAL
        DO 190 I=1,NN
          READ(4,*)N,RHO1(N),U1(N),V1(N),EINT,E1(N),AM
 190    CONTINUE
      ITOTAL=ITOTAL+1
      END IF
      DO 200 I=1,NN
        RHO2(I)=RHO1(I)
        U2(I)=U1(I)
        V2(I)=V1(I)
        E2(I)=E1(I)
 200  CONTINUE
      RETURN
C
C  FORMAT
C
 500  FORMAT(A60)
      END
C
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE BINFO(NN,NBOUN,NB,
     1 UNEW, VNEW, RHONEW, ENEW, UOLD, VOLD, RHOOLD, EOLD)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  THIS SUBROUTINE CALCULATES BOUNDARY INFORMATION
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
```

```
     1  MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
        COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
        DIMENSION MFLAG(2,NDNODE), MFLAGT(2,20), UNORMS(2,NDNODE),
     1  UNORMT(2,20),
     1  UNEW(NDNODE), VNEW(NDNODE), RHONEW(NDNODE), ENEW(NDNODE),
     2  UOLD(NDNODE), VOLD(NDNODE), RHOOLD(NDNODE), EOLD(NDNODE)
C
C  INITIALIZE SOME VARIABLES
C
        DO 150 I=1,NDNODE
          DO 150 J=1,2
  150   UNORMS(J,I)=0.D0
        DO 160 I=1,2
          DO 160 J=1,20
  160   UNORMT(J,I)=0.D0
        DO 175 I=1,NDNODE
  175   MFLAG(2,I)=0
        NB=0
        NC=0
        NCT=0
C
        DO 200 I=1,NBOUN
          NODE1=IBORD(1,I)
          NODE2=IBORD(2,I)
          IFL=IBORD(4,I)
          DELX=COOR(NODE2,1)-COOR(NODE1,1)
          DELY=COOR(NODE2,2)-COOR(NODE1,2)
          D=DSQRT(DELX*DELX+DELY*DELY)
          IF (MFLAG(1,NODE1) .NE. IFL) THEN
            IF (MFLAG(1,NODE1) .EQ. 0) GO TO 240
            NCT=NCT+1
            UNORMT(1,NCT)=DELY/D
            UNORMT(2,NCT)=-DELX/D
            MFLAGT(1,NCT)=NODE1
            MFLAGT(2,NCT)=IFL
            GO TO 250
          END IF
  240     UNORMS(1,NODE1)=UNORMS(1,NODE1)+DELY/D
          UNORMS(2,NODE1)=UNORMS(2,NODE1)-DELX/D
          MFLAG(1,NODE1)=IFL
          MFLAG(2,NODE1)=MFLAG(2,NODE1)+1
  250     IF (MFLAG(1,NODE2) .NE. IFL) THEN
            IF (MFLAG(1,NODE2) .EQ. 0) GO TO 260
            NCT=NCT+1
            UNORMT(1,NCT)=DELY/D
            UNORMT(2,NCT)=-DELX/D
            MFLAGT(1,NCT)=NODE2
            MFLAGT(2,NCT)=IFL
            GO TO 200
          END IF
  260     UNORMS(1,NODE2)=UNORMS(1,NODE2)+DELY/D
          UNORMS(2,NODE2)=UNORMS(2,NODE2)-DELX/D
          MFLAG(1,NODE2)=IFL
          MFLAG(2,NODE2)=MFLAG(2,NODE2)+1
  200   CONTINUE
        DO 300 I=1,NN
          IF (MFLAG(1,I) .NE. 0) THEN
            IF (MFLAG(1,I) .EQ. 1) NEB=NEB+1
            NC=NC+1
            NENODE(1,NC)=I
            NENODE(2,NC)=MFLAG(1,I)
            UNORM(NC,1)=UNORMS(1,I)/REAL(MFLAG(2,I))
            UNORM(NC,2)=UNORMS(2,I)/REAL(MFLAG(2,I))
          END IF
```

```
 300    CONTINUE
        DO 400 I=1,NCT
          N=NC+I
          NENODE(1,N)=MFLAGT(1,I)
          NENODE(2,N)=MFLAGT(2,I)
          UNORM(N,1)=UNORMT(1,I)
          UNORM(N,2)=UNORMT(2,I)
          IF (MFLAGT(1,I) .EQ. 1) NEB=NEB+1
 400    CONTINUE
        NB=NC+NCT
        RETURN
        END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        SUBROUTINE ELEMENT(NE)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES SHAPE FUNCTIONS, THEIR DERIVATIVES AND JACOBIANS
C  AT GAUSSIAN POINTS. ALSO CALCULATES ELEMENT LENGTH VECTORS.
C
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (NDNODE=5000, NDELEM=5000)
C
        COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
        COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1  NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
        DIMENSION G(2), X(4), Y(4), A(2,2), B(3,3)
C
        GOTO (1,2,3) NELTYPE
C
C  SHAPE FUNCTIONS
C
C
C  BILINEAR QUADRILATERALS
C
   1    G(1)=-.57735026918963D0
        G(2)=-G(1)
        W(1)=1.D0
        W(2)=1.D0
        W(3)=1.D0
        W(4)=1.D0
        DO 10 I=1,2
          DO 15 J=1,2
            K=I+I+J-2
            P(1,K)=(1.-G(I))*(1.-G(J))*.25D0
            P(2,K)=(1.+G(I))*(1.-G(J))*.25D0
            P(3,K)=(1.+G(I))*(1.+G(J))*.25D0
            P(4,K)=(1.-G(I))*(1.+G(J))*.25D0
  15      CONTINUE
  10    CONTINUE
        DO 20 I=1,NE
          DO 30 J=1,4
            X(J)=COOR(NO(I,J),1)
  30        Y(J)=COOR(NO(I,J),2)
          X3MX4=X(3)-X(4)
          X3MX2=X(3)-X(2)
          X2MX1=X(2)-X(1)
          X4MX1=X(4)-X(1)
          Y3MY4=Y(3)-Y(4)
          Y3MY2=Y(3)-Y(2)
          Y2MY1=Y(2)-Y(1)
```

```
         Y4MY1=Y(4)-Y(1)
C
C  JACOBIAN
C
         DO 40 J=1,2
           DO 40 K=1,2
             M=J+J+K-2
             A(1,1)=((1.+G(K))*X3MX4+(1.-G(K))*X2MX1)*.25D0
             A(1,2)=((1.+G(K))*Y3MY4+(1.-G(K))*Y2MY1)*.25D0
             A(2,1)=((1.+G(J))*X3MX2+(1.-G(J))*X4MX1)*.25D0
             A(2,2)=((1.+G(J))*Y3MY2+(1.-G(J))*Y4MY1)*.25D0
             DA(I,M)=A(1,1)*A(2,2)-A(1,2)*A(2,1)
C
C  JACOBIAN INVERSE
C
             EJACOB(I,1,1,M)= A(2,2)/DA(I,M)
             EJACOB(I,1,2,M)=-A(1,2)/DA(I,M)
             EJACOB(I,2,1,M)=-A(2,1)/DA(I,M)
             EJACOB(I,2,2,M)= A(1,1)/DA(I,M)
C
C  SHAPE FUNCTION DERIVATIVES
C
             DP(1,M,1,I)=(-A(2,2)*(1.-G(K))+A(1,2)*(1.-G(J)))/DA(I,M)
             DP(1,M,2,I)=( A(2,1)*(1.-G(K))-A(1,1)*(1.-G(J)))/DA(I,M)
             DP(2,M,1,I)=( A(2,2)*(1.-G(K))+A(1,2)*(1.+G(J)))/DA(I,M)
             DP(2,M,2,I)=(-A(2,1)*(1.-G(K))-A(1,1)*(1.+G(J)))/DA(I,M)
             DP(3,M,1,I)=( A(2,2)*(1.+G(K))-A(1,2)*(1.+G(J)))/DA(I,M)
             DP(3,M,2,I)=(-A(2,1)*(1.+G(K))+A(1,1)*(1.+G(J)))/DA(I,M)
             DP(4,M,1,I)=(-A(2,2)*(1.+G(K))-A(1,2)*(1.-G(J)))/DA(I,M)
             DP(4,M,2,I)=( A(2,1)*(1.+G(K))+A(1,1)*(1.-G(J)))/DA(I,M)
             DA(I,M)=DA(I,M)*W(M)
   40      CONTINUE
   20    CONTINUE
         GO TO 1000
C
C  LINEAR/LINEAR TRIANGLES
C
    2    G(1)=1.D0/3.D0
         P(1,1)=1.-G(1)-G(1)
         P(2,1)=G(1)
         P(3,1)=G(1)
         W(1)=1.D0
         DO 120 I=1,NE
           DO 130 J=1,3
             X(J)=COOR(NO(I,J),1)
  130        Y(J)=COOR(NO(I,J),2)
           X2MX1=X(2)-X(1)
           Y2MY1=Y(2)-Y(1)
           X3MX1=X(3)-X(1)
           Y3MY1=Y(3)-Y(1)
C
C  JACOBIAN
C
           A(1,1)=X2MX1
           A(1,2)=Y2MY1
           A(2,1)=X3MX1
           A(2,2)=Y3MY1
           DA(I,1)=A(1,1)*A(2,2)-A(1,2)*A(2,1)
C
C  JACOBIAN INVERSE
C
           EJACOB(I,1,1,1)= A(2,2)/DA(I,1)
           EJACOB(I,1,2,1)=-A(1,2)/DA(I,1)
           EJACOB(I,2,1,1)=-A(2,1)/DA(I,1)
           EJACOB(I,2,2,1)= A(1,1)/DA(I,1)
C
```

```
C  SHAPE FUNCTION DERIVATIVES
C
           DP(1,1,1,I)=(-A(2,2)+A(1,2))/DA(I,1)
           DP(1,1,2,I)=( A(2,1)-A(1,1))/DA(I,1)
           DP(2,1,1,I)=  A(2,2)/DA(I,1)
           DP(2,1,2,I)= -A(2,1)/DA(I,1)
           DP(3,1,1,I)= -A(1,2)/DA(I,1)
           DP(3,1,2,I)=  A(1,1)/DA(I,1)
           DA(I,1)=DA(I,1)*W(1)
  120    CONTINUE
         GO TO 1000
C
C  LINEAR/LINEAR TRIANGLES (THREE POINT INTEGRATION RULE)
C
    3    W(1)=1./3.
         W(2)=1./3.
         W(3)=1./3.
         B(1,1)=.5
         B(2,1)=.5
         B(3,1)=0.
         B(1,2)=0.
         B(2,2)=.5
         B(3,2)=.5
         DO 210 I=1,3
           P(1,I)=1.-B(I,1)-B(I,2)
           P(2,I)=B(I,1)
           P(3,I)=B(I,2)
  210    CONTINUE
         DO 220 I=1,NE
           DO 230 J=1,3
             X(J)=COOR(NO(I,J),1)
  230        Y(J)=COOR(NO(I,J),2)
           X2MX1=X(2)-X(1)
           Y2MY1=Y(2)-Y(1)
           X3MX1=X(3)-X(1)
           Y3MY1=Y(3)-Y(1)
C
C  JACOBIAN
C
           A(1,1)=X2MX1
           A(1,2)=Y2MY1
           A(2,1)=X3MX1
           A(2,2)=Y3MY1
           DO 215 J=1,3
             DA(I,J)=A(1,1)*A(2,2)-A(1,2)*A(2,1)
C
C  JACOBIAN INVERSE
C
             EJACOB(I,1,1,J)= A(2,2)/DA(I,J)
             EJACOB(I,1,2,J)=-A(1,2)/DA(I,J)
             EJACOB(I,2,1,J)=-A(2,1)/DA(I,J)
             EJACOB(I,2,2,J)= A(1,1)/DA(I,J)
C
C  SHAPE FUNCTION DERIVATIVES
C
             DP(1,J,1,I)=(-A(2,2)+A(1,2))/DA(I,J)
             DP(1,J,2,I)=( A(2,1)-A(1,1))/DA(I,J)
             DP(2,J,1,I)=  A(2,2)/DA(I,J)
             DP(2,J,2,I)= -A(2,1)/DA(I,J)
             DP(3,J,1,I)= -A(1,2)/DA(I,J)
             DP(3,J,2,I)=  A(1,1)/DA(I,J)
             DA(I,J)=DA(I,J)*W(J)
  215      CONTINUE
  220    CONTINUE
 1000    RETURN
         END
```

```
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE ASSEM1(NN,NE,RHONEW,UNEW,VNEW,ENEW,F)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES AND ASSEMBLES BOTH SIDES OF THE GLOBAL VECTORS FOR THE
C  NAVIER-STOKES EQUATIONS.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
      DIMENSION RHONEW(NDNODE), UNEW(NDNODE), VNEW(NDNODE),
     1 ENEW(NDNODE)
C
C  UPWIND PARAMETER
C
      ALPHA(X)=1.D0/TANH(X/2.D0)-2.D0/X
C
      DO 5 I=1,NN
        RHR(I)=0.D0
        RHU(I)=0.D0
        RHV(I)=0.D0
        RHE(I)=0.D0
        XLHR(I)=0.D0
        XLHU(I)=0.D0
        XLHV(I)=0.D0
        XLHE(I)=0.D0
    5 CONTINUE
      DO 10 I=1,NE
        DO 20 NG=1,NGAUSS
          RHO=0.
          U=0.
          V=0.
          E=0.
          RHOX=0.
          UX=0.
          VX=0.
          EX=0.
          RHOY=0.
          UY=0.
          VY=0.
          EY=0.
          ZZ=0.
C
C  EVALUATE QUANTITIES AT GAUSSIAN POINTS
C
          DO 30 J=1,NNODE
            NODE=NO(I,J)
```

```
          RHO=RHO+P(J,NG)*RHONEW(NODE)
          U=U+P(J,NG)*UNEW(NODE)
          V=V+P(J,NG)*VNEW(NODE)
          E=E+P(J,NG)*ENEW(NODE)
          RHOX=RHOX+DP(J,NG,1,I)*RHONEW(NODE)
          UX=UX+DP(J,NG,1,I)*UNEW(NODE)
          VX=VX+DP(J,NG,1,I)*VNEW(NODE)
          EX=EX+DP(J,NG,1,I)*ENEW(NODE)
          RHOY=RHOY+DP(J,NG,2,I)*RHONEW(NODE)
          UY=UY+DP(J,NG,2,I)*UNEW(NODE)
          VY=VY+DP(J,NG,2,I)*VNEW(NODE)
          EY=EY+DP(J,NG,2,I)*ENEW(NODE)
          ZZ=ZZ+P(J,NG)
   30   CONTINUE
C
C  PERTURBATION COEFFICIENT FOR THE CONTINUITY EQUATION
C
        U2=U*U
        V2=V*V
        UABS2=U2+V2
        HA=U*EJACOB(I,1,1,NG)+V*EJACOB(I,2,1,NG)
        HB=U*EJACOB(I,1,2,NG)+V*EJACOB(I,2,2,NG)
        H=UABS2/SQRT(HA*HA+HB*HB)*F
        COFC=H*.5/UABS2
C
C  PERTURBATION COEFFICIENT FOR THE X-MOMENTUM EQUATION
C
        UHAT=U*TWOMG
        UHAT2=UHAT*UHAT
        UB2=UHAT2+V2
        HUA=UHAT*EJACOB(I,1,1,NG)+V*EJACOB(I,2,1,NG)
        HUB=UHAT*EJACOB(I,1,2,NG)+V*EJACOB(I,2,2,NG)
        HU=UB2/SQRT(HUA*HUA+HUB*HUB)*F
        GAMU=RHO*RE*HU/(1.+UHAT2/(3.*UB2))
        COFU=ALPHA(GAMU)*HU*.5/UB2
C
C  PERTURBATION COEFFICIENT FOR Y-MOMENTUM EQUATION
C
        VHAT=V*TWOMG
        VHAT2=VHAT*VHAT
        VB2=U2+VHAT2
        HVA=U*EJACOB(I,1,1,NG)+VHAT*EJACOB(I,2,1,NG)
        HVB=U*EJACOB(I,1,2,NG)+VHAT*EJACOB(I,2,2,NG)
        HV=VB2/SQRT(HVA*HVA+HVB*HVB)*F
        GAMV=RHO*RE*HV/(1.+VHAT2/(3.*VB2))
        COFV=ALPHA(GAMV)*HV*.5/VB2
C
C  PERTURBATION COEFFICIENT FOR ENERGY EQUATION
C
        GAME=RHO*PR*RE*H
        COFE=ALPHA(GAME)*H*.5/UABS2
C
C  CALCULATE CONVECTIVE TERMS
C
        RHOCON=U*RHOX+V*RHOY
        UCON=RHO*(UHAT*UX+V*UY)
        VCON=RHO*(U*VX+VHAT*VY)
        ECON=RHO*GAMMA*(U*EX+V*EY)
C
C  CALCULATE FORCING TERMS
C
        EINT=E-C4*UABS2
        UABS2X=U*UX+V*VX
        UABS2Y=U*UY+V*VY
        RHOUX=RHO*UX+U*RHOX
        RHOVY=RHO*VY+V*RHOY
```

```
            RHOFOR=RHO*(UX+VY)
            UFOR=ONEDGM2*(EINT*RHOX+RHO*EX)-GM1*RHO*V*VX
            VFOR=ONEDGM2*(EINT*RHOY+RHO*EY)-GM1*RHO*U*UY
            EFOR=GM1*E*(RHOUX+RHOVY)-
     1           C1*(RHO*(U*UABS2X+V*UABS2Y)+
     2              UABS2*(RHOUX+RHOVY)/2.)
C
C  CALCULATE VISCOUS TERMS
C
            UVIS1=(4.*UX-2.*VY)/3.
            UVIS2=UY+VX
            VVIS1=UVIS2
            VVIS2=(4.*VY-2.*UX)/3.
            EVIS1=U*UVIS1+V*UVIS2
            EVIS2=U*VVIS1+V*VVIS2
C
C  CALCULATE CONDUCTION TERMS
C
            ECON1=EX-C4*2.*UABS2X
            ECON2=EY-C4*2.*UABS2Y
C
C  COMPUTE UPWIND WEIGHTING FUNCTIONS
C
            DO 60 J=1,NNODE
              NODE=NO(I,J)
              UDP=U*DP(J,NG,1,I)
              VDP=V*DP(J,NG,2,I)
              UDPPVDP=UDP+VDP
              PC=P(J,NG)+COFC*UDPPVDP
              PU=P(J,NG)+COFU*(UHAT*DP(J,NG,1,I)+VDP)
              PV=P(J,NG)+COFV*(UDP+VHAT*DP(J,NG,2,I))
              PE=P(J,NG)+COFE*UDPPVDP
C
C  ASSEMBLE RIGHT HAND SIDE (ONLY UPWINDING CONVECTIVE TERMS)
C
              RHR(NODE)=RHR(NODE)-(PC*RHOCON+P(J,NG)*RHOFOR)*DA(I,NG)
              RHU(NODE)=RHU(NODE)-(PU*UCON+P(J,NG)*UFOR+
     1           (DP(J,NG,1,I)*UVIS1+DP(J,NG,2,I)*UVIS2)/RE)*DA(I,NG)
              RHV(NODE)=RHV(NODE)-(PV*VCON+P(J,NG)*VFOR+
     1           (DP(J,NG,1,I)*VVIS1+DP(J,NG,2,I)*VVIS2)/RE)*DA(I,NG)
              RHE(NODE)=RHE(NODE)-(PE*ECON+P(J,NG)*EFOR+
     1           (DP(J,NG,1,I)*EVIS1+DP(J,NG,2,I)*EVIS2)*C2+
     2           (DP(J,NG,1,I)*ECON1+DP(J,NG,2,I)*ECON2)*C3)*DA(I,NG)
C
C  ASSEMBLE LEFT HAND SIDE (LUMPED MASSES)
C
              XLHR(NODE)=XLHR(NODE)+P(J,NG)*ZZ*DA(I,NG)
              XLHU(NODE)=XLHU(NODE)+P(J,NG)*ZZ*RHO*DA(I,NG)
              XLHV(NODE)=XLHV(NODE)+P(J,NG)*ZZ*RHO*DA(I,NG)
              XLHE(NODE)=XLHE(NODE)+P(J,NG)*ZZ*RHO*DA(I,NG)
  60        CONTINUE
  20      CONTINUE
  10    CONTINUE
        RETURN
        END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        SUBROUTINE ASSEM2(NN,NE,RHONEW,UNEW,VNEW,ENEW,F)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES AND ASSEMBLES BOTH SIDES OF THE GLOBAL VECTORS FOR THE
C  EULER EQUATIONS.
C
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
      DIMENSION RHONEW(NDNODE), UNEW(NDNODE), VNEW(NDNODE),
     1 ENEW(NDNODE)
C
C
C
      IF (NELTYPE .EQ. 1) F=2.D0
      IF (NELTYPE .EQ. 2) F=.5D0
      DO 5 I=1,NN
        RHR(I)=0.D0
        RHU(I)=0.D0
        RHV(I)=0.D0
        RHE(I)=0.D0
        XLHR(I)=0.D0
        XLHU(I)=0.D0
        XLHV(I)=0.D0
        XLHE(I)=0.D0
        NMASS(1,I)=1
        DO 5 J=1,20
          XMC(J,I)=0.D0
          XMU(J,I)=0.D0
          XMV(J,I)=0.D0
          XME(J,I)=0.D0
    5 CONTINUE
      DO 10 I=1,NE
        DO 20 NG=1,NGAUSS
          RHO=0.
          U=0.
          V=0.
          E=0.
          RHOX=0.
          UX=0.
          VX=0.
          EX=0.
          RHOY=0.
          UY=0.
          VY=0.
          EY=0.
          ZZ=0.
C
C EVALUATE QUANTITIES AT GAUSSIAN POINTS
C
          DO 30 J=1,NNODE
            NODE=NO(I,J)
            RHO=RHO+P(J,NG)*RHONEW(NODE)
            U=U+P(J,NG)*UNEW(NODE)
            V=V+P(J,NG)*VNEW(NODE)
            E=E+P(J,NG)*ENEW(NODE)
            RHOX=RHOX+DP(J,NG,1,I)*RHONEW(NODE)
            UX=UX+DP(J,NG,1,I)*UNEW(NODE)
            VX=VX+DP(J,NG,1,I)*VNEW(NODE)
```

```
              EX=EX+DP(J,NG,1,I)*ENEW(NODE)
              RHOY=RHOY+DP(J,NG,2,I)*RHONEW(NODE)
              UY=UY+DP(J,NG,2,I)*UNEW(NODE)
              VY=VY+DP(J,NG,2,I)*VNEW(NODE)
              EY=EY+DP(J,NG,2,I)*ENEW(NODE)
              ZZ=ZZ+P(J,NG)
   30       CONTINUE
C
C  PERTURBATION COEFFICIENT FOR THE CONTINUITY EQUATION
C
            U2=U*U
            V2=V*V
            UABS2=U2+V2
            HA=U*EJACOB(I,1,1,NG)+V*EJACOB(I,2,1,NG)
            HB=U*EJACOB(I,1,2,NG)+V*EJACOB(I,2,2,NG)
            H=UABS2/SQRT(HA*HA+HB*HB)*F
            COFC=H*.5/UABS2
C
C  PERTURBATION COEFFICIENT FOR THE X-MOMENTUM EQUATION
C
            UHAT=U*TWOMG
            UHAT2=UHAT*UHAT
            UB2=UHAT2+V2
            HUA=UHAT*EJACOB(I,1,1,NG)+V*EJACOB(I,2,1,NG)
            HUB=UHAT*EJACOB(I,1,2,NG)+V*EJACOB(I,2,2,NG)
            HU=UB2/SQRT(HUA*HUA+HUB*HUB)*F
            COFU=HU*.5/UB2
C
C  PERTURBATION COEFFICIENT FOR Y-MOMENTUM EQUATION
C
            VHAT=V*TWOMG
            VHAT2=VHAT*VHAT
            VB2=U2+VHAT2
            HVA=U*EJACOB(I,1,1,NG)+VHAT*EJACOB(I,2,1,NG)
            HVB=U*EJACOB(I,1,2,NG)+VHAT*EJACOB(I,2,2,NG)
            HV=VB2/SQRT(HVA*HVA+HVB*HVB)*F
            COFV=HV*.5/VB2
C
C  PERTURBATION COEFFICIENT FOR ENERGY EQUATION
C
            COFE=H*.5/UABS2
C
C  CALCULATE CONVECTIVE TERMS
C
            RHOCON=U*RHOX+V*RHOY
            UCON=RHO*(UHAT*UX+V*UY)
            VCON=RHO*(U*VX+VHAT*VY)
            ECON=RHO*GAMMA*(U*EX+V*EY)
C
C  CALCULATE FORCING TERMS
C
            EINT=E-C4*UABS2
            UABS2X=U*UX+V*VX
            UABS2Y=U*UY+V*VY
            RHOUX=RHO*UX+U*RHOX
            RHOVY=RHO*VY+V*RHOY
            RHOFOR=RHO*(UX+VY)
            UFOR=ONEDGM2*(EINT*RHOX+RHO*EX)-GM1*RHO*V*VX
            VFOR=ONEDGM2*(EINT*RHOY+RHO*EY)-GM1*RHO*U*UY
            EFOR=GM1*E*(RHOUX+RHOVY)-
     1          C1*(RHO*(U*UABS2X+V*UABS2Y)+
     2             UABS2*(RHOUX+RHOVY)/2.)
C
C  COMPUTE DISCONTINUITY CAPTURING TERMS IF NEEDED
C
            IF (MUPW .LT. 3) GO TO 39
```

```
            RHOX2=RHOX*RHOX
            RHOY2=RHOY*RHOY
            UX2=UX*UX
            UY2=UY*UY
            VX2=VX*VX
            VY2=VY*VY
            EX2=EX*EX
            EY2=EY*EY
            DRHO2=RHOX2+RHOY2
            DU2=UX2+UY2
            DV2=VX2+VY2
            DE2=EX2+EY2
            DRHO=DSQRT(DRHO2)
            DU=DSQRT(DU2)
            DV=DSQRT(DV2)
            DE=DSQRT(DE2)
C
C  CONTINUITY
C
            IF (DRHO2 .LT. 1.D-5) GO TO 36
            UGR=RHOCON*RHOX/DRHO2
            VGR=RHOCON*RHOY/DRHO2
            UG2=UGR*UGR
            VG2=VGR*VGR
            UABSG2=UG2+VG2
            UABSG=DSQRT(UABSG2)
            UL1=UGR*EJACOB(I,1,1,NG)+VGR*EJACOB(I,2,1,NG)
            UL2=UGR*EJACOB(I,1,2,NG)+VGR*EJACOB(I,2,2,NG)
            ULABS=DSQRT(UL1*UL1+UL2*UL2)
            IF (ULABS .LT. 1.D-10) THEN
              COF1=0.D0
              GO TO 36
            END IF
            HG=UABSG/ULABS*F
            COF1=HG*.5D0/UABSG
C
C  X-MOMENTUM
C
   36       IF (DU2 .LT. 1.D-5) GO TO 37
            UC=U*UX+V*UY
            UGU=UC*UX/DU2
            VGU=UC*UY/DU2
            UG2=UGU*UGU
            VG2=VGU*VGU
            UABSG2=UG2+VG2
            UABSG=DSQRT(UABSG2)
            UL1=UGU*EJACOB(I,1,1,NG)+VGU*EJACOB(I,2,1,NG)
            UL2=UGU*EJACOB(I,1,2,NG)+VGU*EJACOB(I,2,2,NG)
            ULABS=DSQRT(UL1*UL1+UL2*UL2)
            IF (ULABS .LT. 1.D-10) THEN
              COF2=0.D0
              GO TO 37
            END IF
            HG=UABSG/ULABS*F
            COF2=HG*.5D0/UABSG
C
C  Y-MOMENTUM
C
   37       IF (DV2 .LT. 1.D-5) GO TO 38
            VC=U*VX+V*VY
            UGV=VC*VX/DV2
            VGV=VC*VY/DV2
            UG2=UGV*UGV
            VG2=VGV*VGV
            UABSG2=UG2+VG2
            UABSG=DSQRT(UABSG2)
```

```
            UL1=UGV*EJACOB(I,1,1,NG)+VGV*EJACOB(I,2,1,NG)
            UL2=UGV*EJACOB(I,1,2,NG)+VGV*EJACOB(I,2,2,NG)
            ULABS=DSQRT(UL1*UL1+UL2*UL2)
            IF (ULABS .LT. 1.D-10) THEN
              COF3=0.D0
              GO TO 38
            END IF
            HG=UABSG/ULABS*F
            COF3=HG*.5D0/UABSG
C
C   ENERGY
C
  38        IF (DE2 .LT. 1.D-5) GO TO 39
            EC=U*EX+V*EY
            UGE=EC*EX/DE2
            VGE=EC*EY/DE2
            UG2=UGE*UGE
            VG2=VGE*VGE
            UABSG2=UG2+VG2
            UABSG=DSQRT(UABSG2)
            UL1=UGE*EJACOB(I,1,1,NG)+VGE*EJACOB(I,2,1,NG)
            UL2=UGE*EJACOB(I,1,2,NG)+VGE*EJACOB(I,2,2,NG)
            ULABS=DSQRT(UL1*UL1+UL2*UL2)
            IF (ULABS .LT. 1.D-10) THEN
              COF4=0.D0
              GO TO 39
            END IF
            HG=UABSG/ULABS*F
            COF4=HG*.5D0/UABSG
C
C   COMPUTE UPWIND WEIGHTING FUNCTIONS
C
  39        DO 60 J=1,NNODE
              NODE1=NO(I,J)
              UDP=U*DP(J,NG,1,I)
              VDP=V*DP(J,NG,2,I)
              UDPPVDP=UDP+VDP
              PC=P(J,NG)+COFC*UDPPVDP
              PU=P(J,NG)+COFU*(UHAT*DP(J,NG,1,I)+VDP)
              PV=P(J,NG)+COFV*(UDP+VHAT*DP(J,NG,2,I))
              PE=P(J,NG)+COFE*UDPPVDP
              IF (MUPW .EQ. 3) THEN
                UDP=UGR*DP(J,NG,1,I)
                VDP=VGR*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC1=PC+COF1*UDPPVDP
                UDP=UGU*DP(J,NG,1,I)
                VDP=VGU*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC2=PU+COF2*UDPPVDP
                UDP=UGV*DP(J,NG,1,I)
                VDP=VGV*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC3=PV+COF3*UDPPVDP
                UDP=UGE*DP(J,NG,1,I)
                VDP=VGE*DP(J,NG,2,I)
                UDPPVDP=UDP+VDP
                PC4=PE+COF4*UDPPVDP
              END IF
C
C   ASSEMBLE RIGHT HAND SIDE
C
              IF (MUPW .EQ. 1) THEN
                RHR(NODE1)=RHR(NODE1)-(PC*RHOCON+P(J,NG)*RHOFOR)*DA(I,NG)
                RHU(NODE1)=RHU(NODE1)-(PU*UCON+P(J,NG)*UFOR)*DA(I,NG)
                RHV(NODE1)=RHV(NODE1)-(PV*VCON+P(J,NG)*VFOR)*DA(I,NG)
```

```
            RHE(NODE1)=RHE(NODE1)-(PE*ECON+P(J,NG)*EFOR)*DA(I,NG)
          ELSEIF (MUPW .EQ. 2) THEN
            RHR(NODE1)=RHR(NODE1)-PC*(RHOCON+RHOFOR)*DA(I,NG)
            RHU(NODE1)=RHU(NODE1)-PU*(UCON+UFOR)*DA(I,NG)
            RHV(NODE1)=RHV(NODE1)-PV*(VCON+VFOR)*DA(I,NG)
            RHE(NODE1)=RHE(NODE1)-PE*(ECON+EFOR)*DA(I,NG)
          ELSE
            RHR(NODE1)=RHR(NODE1)-PC1*(RHOCON+RHOFOR)*DA(I,NG)
            RHU(NODE1)=RHU(NODE1)-PC2*(UCON+UFOR)*DA(I,NG)
            RHV(NODE1)=RHV(NODE1)-PC3*(VCON+VFOR)*DA(I,NG)
            RHE(NODE1)=RHE(NODE1)-PC4*(ECON+EFOR)*DA(I,NG)
          END IF
C
C  ASSEMBLE LEFT HAND SIDE (CONSISTENT MASSES)
C
          IF (NCON .GT. 1) THEN
            NPASS=0
            DO 70 K=1,NNODE
              NODE2=NO(I,K)
              IF (MUPW .EQ. 1) THEN
                TC=P(J,NG)*P(K,NG)*DA(I,NG)
                TU=P(J,NG)*P(K,NG)*RHO*DA(I,NG)
                TV=TU
                TE=TU
              ELSEIF (MUPW .EQ. 2) THEN
                TC=PC*P(K,NG)*DA(I,NG)
                TU=PU*P(K,NG)*RHO*DA(I,NG)
                TV=PV*P(K,NG)*RHO*DA(I,NG)
                TE=PE*P(K,NG)*RHO*DA(I,NG)
              ELSE
                TC=PC1*P(K,NG)*DA(I,NG)
                TU=PC2*P(K,NG)*RHO*DA(I,NG)
                TV=PC3*P(K,NG)*RHO*DA(I,NG)
                TE=PC4*P(K,NG)*RHO*DA(I,NG)
              END IF
              IF (NODE1 .EQ. NODE2) THEN
                XMC(1,NODE1)=XMC(1,NODE1)+TC
                XMU(1,NODE1)=XMU(1,NODE1)+TU
                XMV(1,NODE1)=XMV(1,NODE1)+TV
                XME(1,NODE1)=XME(1,NODE1)+TE
                GO TO 70
              END IF
              NPASS=NPASS+1
              N1=NMASS(1,NODE1)+NPASS
              XMC(N1,NODE1)=XMC(N1,NODE1)+TC
              XMU(N1,NODE1)=XMU(N1,NODE1)+TU
              XMV(N1,NODE1)=XMV(N1,NODE1)+TV
              XME(N1,NODE1)=XME(N1,NODE1)+TE
              NMASS(N1,NODE1)=NODE2
 70         CONTINUE
          END IF
C
C  LUMPED MASSES
C
        IF (MUPW .EQ. 1) THEN
          XLHR(NODE1)=XLHR(NODE1)+P(J,NG)*ZZ*DA(I,NG)
          XLHU(NODE1)=XLHU(NODE1)+P(J,NG)*ZZ*RHO*DA(I,NG)
          XLHV(NODE1)=XLHU(NODE1)
          XLHE(NODE1)=XLHU(NODE1)
        ELSEIF (MUPW .EQ. 2) THEN
          XLHR(NODE1)=XLHR(NODE1)+PC*ZZ*DA(I,NG)
          XLHU(NODE1)=XLHU(NODE1)+PU*ZZ*RHO*DA(I,NG)
          XLHV(NODE1)=XLHV(NODE1)+PV*ZZ*RHO*DA(I,NG)
          XLHE(NODE1)=XLHE(NODE1)+PE*ZZ*RHO*DA(I,NG)
        ELSE
          XLHR(NODE1)=XLHR(NODE1)+PC1*ZZ*DA(I,NG)
```

```
              XLHU(NODE1)=XLHU(NODE1)+PC2*ZZ*RHO*DA(I,NG)
              XLHV(NODE1)=XLHV(NODE1)+PC3*ZZ*RHO*DA(I,NG)
              XLHE(NODE1)=XLHE(NODE1)+PC4*ZZ*RHO*DA(I,NG)
            END IF
   60       CONTINUE
   20     CONTINUE
          IF (NCON .GT. 1) THEN
            DO 80 J=1,NNODE
              NODE=NO(I,J)
              NMASS(1,NODE)=NMASS(1,NODE)+(NNODE-1)
   80       CONTINUE
          END IF
   10   CONTINUE
        RETURN
        END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        SUBROUTINE SOLVE(NN,NB,
     1                   RHO1,U1,V1,E1,
     2                   RHO2,U2,V2,E2,TIME)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  COMPUTES VALUES OF RHO, U, V AND E AT NEXT TIME STEP
C  AND INCORPORATES THE BOUNDARY CONDITIONS.
C
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (NDNODE=5000, NDELEM=5000)
C
        COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
        COMMON /SOLN/ NMASS(20,NDNODE),
     1  XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2  RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3  XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
        COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1  MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
        COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
        COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
        DIMENSION RHO1(NDNODE), U1(NDNODE), V1(NDNODE), E1(NDNODE),
     1  RHO2(NDNODE), U2(NDNODE), V2(NDNODE), E2(NDNODE), TIME(NDNODE),
     2  DEL0(4,NDNODE), DEL1(4,NDNODE), DEL2(4,NDNODE)
C
C  INCREMENT OF DEPENDENT VARIABLES USING LUMPED MASS
C
        DO 20 I=1,NN
          DEL0(1,I)=TIME(I)*RHR(I)/XLHR(I)
          DEL0(2,I)=TIME(I)*RHU(I)/XLHU(I)
          DEL0(3,I)=TIME(I)*RHV(I)/XLHV(I)
          DEL0(4,I)=TIME(I)*RHE(I)/XLHE(I)
   20   CONTINUE
C
C  LUMPED MASS SOLUTION
C
        IF (NCON .EQ. 1) THEN
          DO 30 I=1,NN
            RHO2(I)=RHO1(I)+DEL0(1,I)
            U2(I)=U1(I)+DEL0(2,I)
            V2(I)=V1(I)+DEL0(3,I)
            E2(I)=E1(I)+DEL0(4,I)
```

```
   30       CONTINUE
            GO TO 100
          END IF
C
C   ITERATIVE SCHEME FOR USE OF CONSISTENT MASS MATRIX
C
          DO 45 I=1,4
            DO 45 J=1,NN
   45     DEL1(I,J)=DEL0(I,J)
          DO 50 ICON=2,NCON
            DO 70 I=1,NN
              DEL2(1,I)=DEL0(1,I)+(1.D0-XMC(1,I)/XLHR(I))*DEL1(1,I)
              DEL2(2,I)=DEL0(2,I)+(1.D0-XMU(1,I)/XLHU(I))*DEL1(2,I)
              DEL2(3,I)=DEL0(3,I)+(1.D0-XMV(1,I)/XLHV(I))*DEL1(3,I)
              DEL2(4,I)=DEL0(4,I)+(1.D0-XME(1,I)/XLHE(I))*DEL1(4,I)
              NUM=NMASS(1,I)
              DO 80 J=2,NUM
                NN=NMASS(J,I)
                DEL2(1,NN)=DEL0(1,NN)-XMC(J,NN)*DEL1(1,NN)
                DEL2(2,NN)=DEL0(2,NN)-XMU(J,NN)*DEL1(2,NN)
                DEL2(3,NN)=DEL0(3,NN)-XMV(J,NN)*DEL1(3,NN)
                DEL2(4,NN)=DEL0(4,NN)-XME(J,NN)*DEL1(4,NN)
   80         CONTINUE
   70       CONTINUE
            IF (ICON .LT. NCON) THEN
              DO 85 I=1,4
                DO 85 J=1,NN
   85         DEL1(I,J)=DEL2(I,J)
            END IF
   50     CONTINUE
          DO 90 I=1,NN
            RHO2(I)=RHO1(I)+DEL2(1,I)
            U2(I)=U1(I)+DEL2(2,I)
            V2(I)=V1(I)+DEL2(3,I)
            E2(I)=E1(I)+DEL2(4,I)
   90     CONTINUE
C
C   ENFORCE BOUNDARY CONDITIONS
C
          CONST=GM1/(2.D0*ONEDGM2)
  100     TW=1.D0
          RI=CINF(1)
          UI=CINF(2)
          VI=CINF(3)
          UI2=UI*UI
          VI2=VI*VI
          UIABS2=UI2+VI2
          ETI=CINF(4)
          EI=ETI-CONST*UIABS2
          PI=CINF(5)
          DO 40 I=1,NB
            NUM=NENODE(1,I)
            RN=RHO1(NUM)
            UN=U1(NUM)
            VN=V1(NUM)
            UN2=UN*UN
            VN2=VN*VN
            UNABS2=UN2+VN2
            ETN=E1(NUM)
            EN=ETN-CONST*UNABS2
            PN=RN*EN
            RP=RHO2(NUM)
            UP=U2(NUM)
            VP=V2(NUM)
            UP2=UP*UP
            VP2=VP*VP
```

```
          UPABS2=UP2+VP2
          ETP=E2(NUM)
          EP=ETP-CONST*UPABS2
          PP=RP*EP
          XN=UNORM(I,1)
          YN=UNORM(I,2)
          IFL=NENODE(2,I)
          GOTO (300,400,500,600) IFL
C
C  "FREE" BOUNDARY
C
 300      SN2=EN/(XMACH*XMACH)
          XM2=UNABS2/SN2
          SN=DSQRT(SN2)
C
C  NORMAL AND TANGENTIAL BOUNDARY VELOCITIES
C
          UNP=UP*XN+VP*YN
          UTP=-UP*YN+VP*XN
          UNN=UN*XN+VN*YN
          UTN=-UN*YN+VN*XN
          UNI=UI*XN+VI*YN
          UTI=-UI*YN+VI*XN
C
C  OUTFLOW
C
          IF (UNP .GE. 0.D0) THEN
C
C  SUPERSONIC
C
            IF (XM2 .GT. 1.D0) GO TO 40
C
C  SUBSONIC
C
            DELP=PI-PP
            RHO2(NUM)=RP+DELP*ONEDGM2/SN2
            UNC=UNP-DELP*ONEDGM2/(RN*SN)
            UTC=UTP
            U2(NUM)=UNC*XN-UTC*YN
            V2(NUM)=UNC*YN+UTC*XN
            UABS2=U2(NUM)*U2(NUM)+V2(NUM)*V2(NUM)
            E2(NUM)=PI*ONEDGM2/RHO2(NUM)+CONST*UABS2
            GO TO 40
          END IF
C
C  INFLOW
C
          IF (ITYPE .EQ. 0 .OR. XM2 .GT. 1.D0) THEN
C
C  NAVIER-STOKES OR SUPERSONIC EULER
C
            RHO2(NUM)=RI
            U2(NUM)=UI
            V2(NUM)=VI
            E2(NUM)=ETI
            GO TO 40
          END IF
C
C  SUBSONIC EULER
C
          PC=0.5D0*(PI+PP+RN*SN*(-UNI+UNP)/ONEDGM2)
          RC=RI+(PC-PI)*ONEDGM2/SN2
          UNC=UNI-(PI-PC)*ONEDGM2/(RN*SN)
          UTC=UTI
          RHO2(NUM)=RC
          U2(NUM)=UNC*XN-UTC*YN
```

```
          V2(NUM)=UNC*YN+UTC*XN
          UABS2=U2(NUM)*U2(NUM)+V2(NUM)*V2(NUM)
          E2(NUM)=PC/RC+CONST*UABS2
          GO TO 40
C
C  ADIABATIC WALL
C
 400      IF (ITYPE .EQ. 1) THEN
C
C  EULER
C
            U2(NUM)=UP*(1.0D0-XN*XN)-VP*XN*YN
            V2(NUM)=VP*(1.0D0-YN*YN)-UP*XN*YN
            GO TO 40
          END IF
C
C  NAVIER-STOKES
C
          U2(NUM)=0.D0
          V2(NUM)=0.D0
          GO TO 40
C
C  CONSTANT TEMPERATURE WALL
C
 500      IF (ITYPE .EQ. 1) THEN
C
C  EULER
C
            U2(NUM)=UP*(1.0D0-XN*XN)-VP*XN*YN
            V2(NUM)=VP*(1.0D0-YN*YN)-UP*XN*YN
            UABS2=U2(NUM)*U2(NUM)+V2(NUM)*V2(NUM)
            E2(NUM)=TW+CONST*UABS2
            GO TO 40
          END IF
C
C  NAVIER-STOKES
C
          U2(NUM)=0.D0
          V2(NUM)=0.D0
          E2(NUM)=TW
          GO TO 40
C
C  SYMMETRY BOUNDARY
C
 600      U2(NUM)=UP*(1.0D0-XN*XN)-VP*XN*YN
          V2(NUM)=VP*(1.0D0-YN*YN)-UP*XN*YN
  40  CONTINUE
      RETURN
      END
C
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE STEADY(NN,ITOTAL,NEXT,CONTOT)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CHECK FOR STEADY STATE AND REASSIGN AND ASSIGN VALUE OF 1 TO
C  VARIABLE NEXT IF CONVERGENCE CRITERION IS MET.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
```

```
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
       DIMENSION CON(4)
C
       EPS=1.D-4
       DO 10 I=1,4
         CON(I)=0.D0
  10   CONTINUE
       DO 20 I=1,NN
         RR=RHR(I)*RHR(I)
         UU=RHU(I)*RHU(I)
         VV=RHV(I)*RHV(I)
         EE=RHE(I)*RHE(I)
         CON(1)=CON(1)+RR
         CON(2)=CON(2)+UU
         CON(3)=CON(3)+VV
         CON(4)=CON(4)+EE
  20   CONTINUE
       TOTAL=CON(1)+CON(2)+CON(3)+CON(4)
       DEL=(TOTAL-CONTOT)/TOTAL
       WRITE(6,100)ITOTAL,(CON(I),I=1,4),TOTAL
       IF (DABS(DEL) .LT. EPS) THEN
         NEXT=1
         RETURN
       END IF
       CONTOT=TOTAL
       RETURN
C
C  FORMAT
C
 100   FORMAT (I5,5(1X,E9.4))
       END
C
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
       SUBROUTINE TSTEP1(NN,NE,UNEW,VNEW,ENEW,DELTP,F)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES TIME STEP FOR NAVIER-STOKES SOLVER.
C
       IMPLICIT REAL*8 (A-H,O-Z)
       PARAMETER (NDNODE=5000, NDELEM=5000)
C
       COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
       COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
       COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
       COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
       COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
       DIMENSION UNEW(NDNODE), VNEW(NDNODE), ENEW(NDNODE),
     1 DELTP(NDNODE), DELTE(NDELEM)
C
       D=REAL(NNODE)
       DO 10 I=1,NE
         U=0.
```

```
        V=0.
        E=0.
        DO 20 J=1,NNODE
          NODE=NO(I,J)
          U=U+UNEW(NODE)
          V=V+VNEW(NODE)
          E=E+ENEW(NODE)
   20   CONTINUE
        U=U/D
        V=V/D
        E=E/D
        U2=U*U
        V2=V*V
        UABS2=U2+V2
        TEMP=E-C4*UABS2
        HA1=U*EJACOB(I,1,1,1)+V*EJACOB(I,2,1,1)
        HB1=U*EJACOB(I,1,2,1)+V*EJACOB(I,2,2,1)
        H1=UABS2/SQRT(HA1*HA1+HB1*HB1)*F
        HA2=V*EJACOB(I,1,1,1)+U*EJACOB(I,2,1,1)
        HB2=V*EJACOB(I,1,2,1)+U*EJACOB(I,2,2,1)
        H2=UABS2/SQRT(HA2*HA2+HB2*HB2)*F
        TERM1=GAMMA*UABS2
        TERM2=SQRT(TEMP*UABS2)/XMACH
        TERM3=2.*C3*UABS2/H1
        TERM4=2.*C3*UABS2*H1/(H2*H2)
        DELTE(I)=SAFE*H1/(TERM1+TERM2+TERM3+TERM4)
   10 CONTINUE
      NPASS=0
 3000 NPASS=NPASS+1
      DO 100 I=1,NN
  100 DELTP(I)=1.D6
      DO 5000 IE=1,NE
        DO 4000 IN=1,NNODE
          IP=NO(IE,IN)
          DELTP(IP)=MIN(DELTP(IP),DELTE(IN))
 4000   CONTINUE
 5000 CONTINUE
C
      DO 7000 IE=1,NE
      KOUNT=0
      DO 6000 IN=1,NNODE
        IP=NO(IE,IN)
        TOLER=0.8D0*DELTE(IE)-DELTP(IP)
        IF (TOLER .GT. 1.D-4) THEN
          KOUNT=KOUNT+1
          DELTE(IE)=DELTP(IP)/0.8D0
        END IF
 6000   CONTINUE
 7000 CONTINUE
C
      IF (NPASS .GE. 10) GO TO 1000
      IF (KOUNT .NE. 0) GO TO 3000
 1000 RETURN
      END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      SUBROUTINE TSTEP2(NN,NE,UNEW,VNEW,ENEW,DELTP,F)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  CALCULATES TIME STEP FOR EULER SOLVER.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NDNODE=5000, NDELEM=5000)
```

```
C
      COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
      COMMON /SOLN/ NMASS(20,NDNODE),
     1 XLHR(NDNODE), XLHU(NDNODE), XLHV(NDNODE), XLHE(NDNODE),
     2 RHR(NDNODE), RHU(NDNODE), RHV(NDNODE), RHE(NDNODE),
     3 XMC(20,NDNODE), XMU(20,NDNODE), XMV(20,NDNODE), XME(20,NDNODE)
C
      COMMON /SHAPE/ P(4,4), DP(4,4,2,NDELEM), DA(NDELEM,4),
     1 NELTYPE, NGAUSS, NNODE, EJACOB(NDELEM,2,2,4), W(4)
C
      COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1 MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
      COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
      DIMENSION UNEW(NDNODE), VNEW(NDNODE), ENEW(NDNODE),
     1 DELTP(NDNODE), DELTE(NDELEM)
C
      D=REAL(NNODE)
      DO 10 I=1,NE
        U=0.
        V=0.
        E=0.
        DO 20 J=1,NNODE
          NODE=NO(I,J)
          U=U+UNEW(NODE)
          V=V+VNEW(NODE)
          E=E+ENEW(NODE)
   20   CONTINUE
        U=U/D
        V=V/D
        E=E/D
        U2=U*U
        V2=V*V
        UABS2=U2+V2
        TEMP=E-C4*UABS2
        HA=U*EJACOB(I,1,1,1)+V*EJACOB(I,2,1,1)
        HB=U*EJACOB(I,1,2,1)+V*EJACOB(I,2,2,1)
        H=UABS2/SQRT(HA*HA+HB*HB)*F
        TERM1=GAMMA*UABS2
        TERM2=SQRT(TEMP*UABS2)/XMACH
        DELTE(I)=SAFE*H/(TERM1+TERM2)
   10 CONTINUE
      NPASS=0
 3000 NPASS=NPASS+1
      DO 100 I=1,NN
  100 DELTP(I)=1.D6
      DO 5000 IE=1,NE
        DO 4000 IN=1,NNODE
          IP=NO(IE,IN)
          DELTP(IP)=MIN(DELTP(IP),DELTE(IN))
 4000   CONTINUE
 5000 CONTINUE
      IF (NTST .EQ. 1) GO TO 4050
      T=1.D4
      DO 5050 I=1,NN
 5050 T=MIN(T,DELTP(I))
      DO 5060 I=1,NN
 5060 DELTP(I)=T
      GO TO 1000
C
C  LOCAL TIMESTEPS
C
 4050 DO 7000 IE=1,NE
      KOUNT=0
```

```
        DO 6000 IN=1,NNODE
          IP=NO(IE,IN)
          TOLER=0.8D0*DELTE(IE)-DELTP(IP)
          IF (TOLER .GT. 1.D-4) THEN
            KOUNT=KOUNT+1
            DELTE(IE)=DELTP(IP)/0.8D0
          END IF
 6000     CONTINUE
 7000   CONTINUE
C
        IF (NPASS .GE. 10) THEN
          WRITE(6,*)' NPASS > 10 IN TSTEP2'
          STOP
        END IF
        IF (KOUNT .NE. 0) GO TO 3000
 1000   RETURN
        END
C
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        SUBROUTINE DOUTPT(NN,TITLE,ITOTAL,RHO,U,V,E)
C
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C  PRINTS DEPENDENT VARIABLES TO RESTART FILE
C
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (NDNODE=5000, NDELEM=5000)
C
        COMMON /GEOM/ COOR(NDNODE,2), NO(NDELEM,4), UNORM(NDNODE,2)
C
        COMMON /PARAM/ RE, PR, GAMMA, XMACH, ALF,
     1  MUPW, NFLOW, NCON, NTST, ITYPE, SAFE, ISTART
C
        COMMON /CONST/ GM1, TWOMG, ONEDGM2, C1, C2, C3, C4, C5
C
        COMMON /BOUND/ NENODE(2,NDNODE), IBORD(4,NDNODE), CINF(5)
C
        DIMENSION U(NDNODE), V(NDNODE), RHO(NDNODE), E(NDNODE)
C
C       CHARACTER TITLE*60
C
C  WRITE TITLE AND NUMBER OF TIME STEPS
C
        WRITE(8,500)TITLE
        WRITE(8,*)ITOTAL
C
C  OUTPUT QUANTITIES
C
        CONST=GM1/(2.D0*ONEDGM2)
        DO 100 I=1,NN
          U2=U(I)*U(I)
          V2=V(I)*V(I)
          EINT=E(I)-CONST*(U2+V2)
          XM=XMACH*SQRT((U2+V2)/EINT)
          WRITE(8,560) I, RHO(I), U(I), V(I), EINT, E(I), XM
  100   CONTINUE
C
C  FORMATS
C
  500   FORMAT (A60)
  560   FORMAT (I5,6(1X,F9.5))
        END
```