A Finite Element Solver for
3-D Compressible Viscous Flows

K. C. Reddy, J. N. Reddy and S. Nayani

The University of Tennessee Space Institute
Tullahoma, Tennessee 37388

Final Report of Contract No. NAS8-36555

Submitted to
NASA/MSFC
Marshall Space Flight Center, AL 35812

by
The University of Tennessee Space Institute
Tullahoma, Tennessee 37388

January 1990
Table of Contents

1. Introduction .................................. 1

2. Locally Implicit Scheme for a Model Equation .......... 4

3. Locally Implicit Scheme for Navier-Stokes Equations ...... 8
   3.1 Finite Element Approximations .................... 8
   3.2 Locally Implicit Scheme .......................... 10
   3.3 Surface Flux Computation ........................ 12
   3.4 Artificial Dissipation ........................... 13

4. Test Calculations ................................ 14
   4.1 Couette Flow .................................. 14
   4.2 Laminar Boundary Layer Over a Flat Plate .......... 14
   4.3 Flow Over an Airfoil ........................... 14
   4.4 Flow in a Turn-around Duct ...................... 15

5. References .................................... 16

Figures ......................................... 17

Appendix I ....................................... 31
Appendix II ...................................... 33
Appendix III ..................................... 38
1. Introduction

Computation of the flow field inside a space shuttle main engine (SSME) requires the application of the state-of-the-art CFD technology. Several computer codes are under development to solve three dimensional Navier–Stokes equations for analyzing the SSME internal flow, such as the flow through the hot gas manifold. The computational methods used in the Navier–Stokes codes fall into two major categories: finite difference and finite element methods. Some of the algorithms are designed to solve the unsteady compressible Navier–Stokes equations, either by explicit or by implicit factorization methods, using several hundred or thousands of time steps to reach a steady-state solution asymptotically. Other algorithms attempt to solve the steady-state equations by relaxation methods. All of them require body-fitting curvilinear grids with sufficient resolution. Grid requirements, however, differ greatly with the region being modelled and the algorithm used. Implicit factorization based on finite differences typically uses global numerical transformations whereby the transformed grid in the computational space is uniform and rectilinear. This requires the grid to have indices which are separable in the three directions for three dimensional problems, and also be reasonably smooth. However, such requirements may introduce grid singularities when complicated domains are discretized. Flow solver algorithm will have to deal with such grid singularities. Explicit schemes and finite element algorithms have less stringent requirements on the grid structure. However, explicit schemes are slow to converge because of the stability limitations on time step, particularly for large scale viscous problems.

The finite element method is characterized by three basic features which are credited for the enormous success the method has enjoyed in the solution of practical engineering problems. The first feature is that every computational domain is viewed as a collection of simple subdomains, called finite elements. This feature allows us to represent complicated geometries as assemblages of simple parts. It is a desirable feature in the solution of flow problems in complex configurations, not only to describe the complex geometry but also to choose the most suitable computational grid for a particular flow. This feature also allows us to place or remove any obstructions routinely into the flow field. The second feature is that over each element the solution is represented by polynomials of desired degree. This allows us to compute the solution as a continuous function of position instead of at selected few points. The third feature is that the relationship (i.e., the algebraic equations) between the solution and its dual variables is developed using a variational method, such as the Galerkin method. The boundary conditions are then applied on the algebraic equations directly before solving. The three features of the finite element method also allow the easy development and interfacing of pre- and post-processors, and user-defined subroutines for equations for state and turbulence models.

The Galerkin finite element method (i.e., the weight functions are the same as the approximation functions) applied to flow problems always results in implicit schemes. The
weighted-residual (or Petrov–Galerkin) method, in which the weight functions are different from the approximation functions, can be used in conjunction with explicit schemes to obtain explicit final equations. For example, by selecting the weight functions to be orthogonal to the approximation functions, the mass matrix can be diagonalized. However, such considerations are entirely in the interest of obtaining explicit schemes and not necessarily in the interest of accuracy or even computational efficiency. In the current project an implicit finite element scheme with suitable dissipation terms for stability is developed. A relaxation procedure, known as the locally implicit scheme is developed to solve the coupled set of algebraic equations efficiently.

Allowing the possibility of unstructured grids is important for discretizing complex flow domains efficiently and also for adding the features of solution-adaptive grids. For grids with large numbers of nodes, direct solution procedures for the finite element equations become impractical. Thus we have undertaken the development of a new iterative algorithm for the solution of implicit finite element equations without assembling global matrices. It is an efficient iteration scheme based on a modified non-linear Gauss–Seidel iteration with symmetric sweeps. This algorithm is analyzed for a model equation and is shown to be unconditionally stable. This analysis is reported in the next Section.

The locally implicit scheme is unconditionally stable based on local linearized analysis. However, for strongly convective flows there is a possibility of non-linear numerical instabilities occurring in some parts of the flow domain and eventually destabilizing the entire flow domain. We have added adaptive artificial dissipation terms of third order to the finite element approximations similar to Jameson and others\(^1\). These are designed to suppress non-linear instabilities if they appear and at the same time be much smaller than the real viscosity terms in viscous zones.

In numerical schemes for solving fluid flow equations, there is some degree of uncertainty as to the imposition of boundary conditions on some of the variables at different types of boundaries, particularly at the inflow and outflow boundaries. In the current finite element code we have developed special procedures to compute the required flux terms at the boundary surfaces to the same degrees of accuracy as in the interior. We expect that our technique of computing the required surface fluxes iteratively, together with the interior flow variables, should minimize the uncertainties in the imposition of boundary conditions.

The locally implicit scheme is tested on a variety of problems. It has been shown to be efficient with multi-grid acceleration procedures for elliptic problems by Reddy and Nayani\(^2\) and for inviscid compressible flows from transonic to supersonic Mach numbers by Reddy and Jacocks\(^3\). Reddy, Reddy and Nayani\(^4\) have developed this scheme for viscous flow problems. We developed a 2–D test code for solving unsteady compressible Navier–Stokes equations with finite volume approximation, which is a special case of the finite element approximation. This code has been used to check various features of the
locally implicit solution algorithm. We have also added an algebraic turbulence model developed by Baldwin and Lomax\textsuperscript{(5)}.

Results for a series of test problems are presented in this report. The finite element code has been tested for Couette flow, described in Schlichting\textsuperscript{(6)}, which is a flow under a pressure gradient between two parallel plates in relative motion. Another problem that has been solved is viscous laminar flow over a flat plate. As a test case for the locally implicit scheme, the 2-D finite volume code has been applied to compute subsonic and transonic viscous flows over airfoils for both laminar and turbulent cases. The general 3-D finite element code has been used to compute the flow in an axisymmetric turnaround duct at low Mach numbers.
2. Locally Implicit Scheme for a Model Equation

Locally implicit scheme is a relaxation method for solving the non-linear finite element equations approximating the Navier–Stokes equations. It is a point iteration method at each time step. However, it is not necessary for the iteration to converge fully at each time step if we are interested in computing the time asymptotic steady-state solutions. The analysis of the consistency, stability and hence convergence of the scheme is presented for a model equation for the Navier–Stokes equations.

Consider a one-dimensional convection-diffusion equation,

\[
\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}
\]  

(2.1)

Finite element approximation at a node \( j \) on a uniform mesh for equation (2.1) can be written as

\[
\frac{\partial}{\partial t} \int u \phi_j dx + \int \left( -au + \nu \frac{\partial u}{\partial x} \right) \frac{\partial \phi_j}{\partial x} dx = 0
\]

(2.2)

where \( \phi_j \) is a global test function corresponding to the node \( j \). For a linear element approximation, equation (2.2) gives

\[
\frac{\partial}{\partial t} \left\{ \frac{1}{6} u_{j-1} + \frac{2}{3} u_j + \frac{1}{6} u_{j+1} \right\} + \left( \frac{a}{2\Delta x} \right) (u_{j+1} - u_{j-1})
\]

\[- \left( \frac{\nu}{\Delta x^2} \right) (u_{j-1} - 2u_j + u_{j+1}) = 0
\]

(2.3)

Implicit time integration gives

\[
\frac{1}{6} \Delta u_{j-1} + \frac{2}{3} \Delta u_j + \frac{1}{6} \Delta u_{j+1} + C \left( u_{j+1}^{n+1} - u_{j-1}^{n+1} \right)
\]

\[- R \left( u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1} \right) = 0
\]

(2.4)

where \( \Delta u_j = u_j^{n+1} - u_j^n \)

\[C = a\Delta t/\Delta x, \quad R = \nu\Delta t/\Delta x^2\]

Equation (2.4), together with appropriate boundary conditions, gives a system of linear equations which can be solved easily in one-dimension and this scheme is unconditionally stable. However, the system of equations becomes too large in multi-dimensions and various types of sparse matrix solvers are developed in the literature, but they are usable only with a modest number of nodes. Alternately, we develop a relaxation scheme to solve (2.4) approximately at each time step. The scheme is a modification of the symmetric Gauss–Seidel iteration. The basic Gauss–Seidel iteration, even with symmetric sweeps, is unstable
for a whole range of Courant number \( C \) in equation (2.4). The present modification makes it unconditionally stable. Rewrite the equation (2.4) in delta form as

\[
\frac{1}{6} \Delta u_{j-1} + \frac{2}{3} \Delta u_j + \frac{1}{6} \Delta u_{j+1} + \frac{C}{2} (\Delta u_{j+1} - \Delta u_{j-1}) - R (\Delta u_{j-1} - 2\Delta u_j + \Delta u_{j+1}) = \text{Res}_j^n
\]

(2.5)

where

\[
\text{Res}_j^n = -\frac{C}{2} (u_{j+1}^n - u_{j-1}^n) + R (u_{j-1}^n - 2u_j^n + u_{j+1}^n)
\]

(2.6)

As \( \Delta u_j = u_{j+1}^n - u_j^n \to 0 \) as \( n \to \infty \), we obtain the asymptotic steady-state solution as the \( \text{Res}_j \) function is driven to zero. This process may be speeded up and made more robust by choosing a value for \( R \) on the left side of equation (2.5) larger than the value of \( R \) on the right side of equation (2.5). To analyze this process we use the notation \( \overline{R} \) for \( R \) on the left side of equation (2.5). It may be noted that we can always obtain time accurate solution, if that is required, by choosing \( \overline{R} = R \). We solve for \( \Delta u_j \) at each time step by a modified Gauss-Seidel iteration:

\[
\Delta u_j^{(m+1)} = \Delta u_j^{(m)} + du_j, \quad \Delta u_j^{(0)} = 0
\]

(2.7)

Left-to-right sweep yields

\[
\frac{2}{3} du_j + \frac{1}{6} du_{j+1} + \frac{C}{2} du_{j+1} - \overline{R} (-2du_j + du_{j+1}) = \text{RHS}
\]

(2.8)

where

\[
\text{RHS} = \text{Res}_j^n - \left[ \frac{1}{6} \Delta u_{j-1}^{(m+1)} + \frac{2}{3} \Delta u_j^{(m)} + \frac{1}{6} \Delta u_{j+1}^{(m)} + \frac{C}{2} (\Delta u_{j+1}^{(m)} - \Delta u_{j-1}^{(m+1)}) - \overline{R} \left( \Delta u_{j-1}^{(m+1)} - 2\Delta u_j^{(m)} + \Delta u_{j+1}^{(m)} \right) \right]
\]

(2.9)

Now we approximate \( du_{j+1} \sim du_j \) and replace \( C \) by its absolute value \( |C| \) on the left side of equation (2.8), to accommodate convection velocity direction either in or opposite to the iteration sweep direction. This leads to an explicit expression for \( du_j \):

\[
\left( \frac{5}{6} + \frac{|C|}{2} + \overline{R} \right) du_j = \text{RHS}
\]

(2.10)

Right-to-left sweep is defined similarly. A symmetric iteration sweep consists of a left-to-right sweep followed by a right-to-left sweep. It may be noted that \( du_j \) is the iterative correction to the time change iterates \( \Delta u_j^{(m)} \) and if the iteration process is convergent,
RHS \to 0 and the equation (2.5) can be satisfied as accurately as we wish by carrying out the necessary number of symmetric iteration sweeps. The approximations made in the iteration do not affect the actual solution itself. Thus the iteration equations are consistent with the basic equations. One or two symmetric sweeps per time step are usually sufficient for obtaining steady-state solutions. Local stability analysis can be carried out by computing the amplification factor of discrete Fourier modal solutions per time step. In this analysis, we seek modal solutions of the equations (2.9) and (2.10) in the form

\[ u^n_j = v^n e^{ij\xi}, \quad 0 \leq \xi = \alpha \Delta x \leq \pi \]
\[ \Delta u^{(m)}_j = \Delta v^{(m)} e^{ij\xi}, \quad m = 0, 1, \ldots \]
\[ u^{n+1}_j = v^{n+1} e^{ij\xi} \]

For a single symmetric sweep per time step \((m = 0, 1)\),

\[ v^{n+1} = v^n + \Delta v^{(2)} = g(\xi)v^n \]

where \(g(\xi)\) is known as the amplification factor from one time step to the next and is given by

\[ g(\xi) = 1 + \frac{r}{h_3} \left[ 1 + \frac{h_2}{h_1} \right], \quad 0 \leq \xi \leq \pi \]
\[ r = -Ci \sin \xi + 2R(\cos \xi - 1) \]
\[ h_1 = b - e^{-i\xi} \left( \frac{C}{2} + \overline{R} - \frac{1}{6} \right) \]
\[ h_2 = b - \frac{2}{3} - 2\overline{R} + e^{-i\xi} \left( \frac{C}{2} + \overline{R} - \frac{1}{6} \right) \]
\[ h_3 = b + e^{i\xi} \left( \frac{C}{2} - \overline{R} + \frac{1}{6} \right) \]
\[ b = \frac{5}{6} + \frac{|C|}{2} + \overline{R} \]

A necessary condition for stability is \(|g(\xi)| \leq 1\). It can be shown that \(|g(\xi)|\) is indeed \(\leq 1\) unconditionally. It is also desirable to have \(|g(\xi)| < 1\) as much as possible for \(\xi\) closer to \(\pi\) which represents the range of high frequency modes of the solution. Figure 1 shows plots of \(|g(\xi)|\) versus \(\xi\) for different Courant numbers for \(R = \overline{R} = \frac{C}{64}\). Figure 2 shows plots of \(|g|\) versus \(\xi\) for \(C = 10, \overline{R} = R\) and \(R\) takes different values. Figure 3 shows the plots for \(C = 10, \overline{R} = 2R\) and \(R\) takes different values. Numerical plots of \(|g|\) against \(\xi\) confirm that the scheme is unconditionally stable. However, very large Courant numbers are not necessarily the best. Courant number \(C \simeq 10\) and \(\overline{R} = 2R \to 4R\) seem desirable ranges. Amplification factors corresponding to two or more symmetric modified Gauss-Seidel iterations have similar behavior. Thus we establish unconditional stability for the modified Gauss-Seidel iteration scheme for the convection-diffusion equation. Similar stability can be shown.
when the diffusion term is replaced by a 4th difference term of the type that is used as artificial viscosity term of third order for suppressing non-linear instabilities for convection dominated flows. It is possible to use artificial viscosity terms which are smaller than the truncation terms of the second order accurate finite element approximations. In the present Navier-Stokes finite element code where we compute all terms to full second order accuracy, artificial dissipation terms, which are an order of magnitude smaller than truncation error, are included to suppress non-linear instabilities. Stability analysis of the model equation indicates that the locally implicit scheme is unconditionally stable in a local linearized sense.
3. Locally Implicit Scheme for Navier–Stokes Equations

Many algorithms designed to solve the unsteady compressible Navier-Stokes equations use either explicit methods or implicit factorization methods. Finite element approximations usually yield implicit equations. These are solved by explicit time integration methods after making additional approximations. Explicit methods may take thousands of time steps to converge. Solving them implicitly with Newton iteration is possible, but the matrix storage requirements for the resulting algebraic equations and the solution process make it prohibitive even for modest size three dimensional flow problems. There are other algorithms based on relaxation methods. We have developed a locally implicit method for solving the non-linear finite element approximations for 3-D Navier-Stokes equations at each time step.

The method is based on a relaxation procedure for solving the finite element equations corresponding to each node iteratively. The equations for the elements surrounding a particular node are evaluated based on the latest iterates for the flow variables at the nodes around it and the solution is updated at that node by a modified Gauss-Seidel iteration. This procedure does not require the assembly of a global matrix, in contrast to the standard finite element algorithms. It does not require the solution of a system of large number of equations. Thus it is a matrix-free implicit finite element algorithm. An additional feature of the algorithm is that while it uses tri-linear approximations for the flow variables in quadilateral (brick) elements, all the non-linear fluxes in the Navier-Stokes equations are evaluated without any further linear approximation. The fluxes are non-linear and are computed accordingly. This assures the second order spatial accuracy of the scheme even for unstructured grids.

3.1 Finite Element Approximations

The unsteady, compressible Navier-Stokes equations are written in conservation form as

\[
\left\{ \frac{\partial U}{\partial t} \right\} + \nabla \cdot \{ F^v \} + \nabla \cdot \{ F^I \} = \{ 0 \}
\]  

(3.1)

where

\[
\{ U \} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho \epsilon \end{bmatrix}, \quad \{ F^v \} = \begin{bmatrix} \bar{q} \\ -\bar{I} \\ -\bar{I} \cdot \bar{v} + g \end{bmatrix}, \quad \{ F^I \} = \begin{bmatrix} \rho \bar{v} \\ \rho \bar{v} \cdot \bar{v} + p \bar{I} \\ \bar{v} (\rho \epsilon + p) \end{bmatrix}
\]

\{ F^I \} and \{ F^v \} represent the inviscid and viscous fluxes respectively. Details of these equations are given in Appendix I.
The variational form (weak form) of equation (3.1) over an element \( \Omega^e \) is written as

\[
0 = \int_{\Omega^e} \left( \{\Phi\}^T \left\{ \frac{\partial U}{\partial t} \right\} - \{\nabla\Phi\}^T \cdot \{\vec{F}^v + \vec{F}^f\} \right) dV + \oint_{S^e} \{\Phi\}^T \{F_n\} dS \tag{3.2}
\]

where \( \{\Phi\} \) are test functions. They are tri-linear functions for linear finite element approximation and piecewise constants for finite volume approximations. \( F_n = (\vec{F}^v + \vec{F}^f) \cdot \vec{n} \) where \( \vec{n} \) is the outward drawn unit normal to the surface \( S^e \) of the element \( \Omega^e \). The conservation variables \( \vec{U} = (U_\alpha, \alpha = 1, \ldots, 5) \) are approximated by the interpolation functions \( \Psi_j \) as

\[
U_\alpha = \sum_{j=1}^{N} \hat{U}^j_\alpha \Psi_j(x, y, z) \equiv \{\Psi\}\{\hat{U}_\alpha\} \tag{3.3}
\]

where

\[
\{\Psi\} = \{\psi_1, \psi_2, \ldots, \psi_N\}, \{\hat{U}_\alpha\} = \left( \hat{U}^1_\alpha, \hat{U}^2_\alpha, \ldots, \hat{U}^N_\alpha \right)^T.
\]

\( \hat{U}^j_\alpha \) is the numerical value of the \( \alpha \)th component of the conservation flow variable \( U \) at \( j \)th local node of the element \( \Omega^e \). The interpolation functions \( \hat{\Psi} \) and test functions \( \Phi \) are chosen to be the same for compressible flow equations. \( N = 8 \) for tri-linear approximations on quadrilateral brick elements. These approximations are done according to the standard finite element approximations (Ref. 7).

Define the total nodal vector of the conservation variables at the nodes of an element as

\[
\{\hat{U}\} = \begin{bmatrix} \{\hat{U}_1\} \\ \{\hat{U}_2\} \\ \vdots \\ \{\hat{U}_5\} \end{bmatrix}, \quad [\Psi]^e = \begin{bmatrix} \{\Psi\} & \{0\} & \{0\} & \{0\} & \{0\} \\ \{0\} & \{\Psi\} & \{0\} & \{0\} & \{0\} \\ \{0\} & \{0\} & \{\Psi\} & \{0\} & \{0\} \\ \{0\} & \{0\} & \{0\} & \{\Psi\} & \{0\} \\ \{0\} & \{0\} & \{0\} & \{0\} & \{\Psi\} \end{bmatrix} \tag{3.4}
\]

Then

\[
\{U\} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_5 \end{bmatrix} = [\Psi]^e \{\hat{U}\}^e
\]

Now the variational statement (2) can be written as

\[
\{0\} = \int_{\Omega^e} \left( [\Psi]^T [\Psi] \{\hat{U}\} - [\nabla \Psi]^T \cdot \{\vec{F}\} \right) dV + \oint_{S^e} [\Psi]^T \{F_n\} dS \tag{3.5}
\]

It should be noted at this point that \( \vec{F} \) and \( F_n \) are non-linear functions of \( \hat{U} \) and thus the integrals involving them can be expressed analytically in terms of the components of \( \hat{U} \). These expressions are long but they can be programmed into the computer code.
efficiently. The coupled non-linear differential equations (3.5) are discretized in time by the Euler implicit scheme as follows:

\[ \frac{1}{\Delta t} [M^e] \{ \Delta \bar{U}^e \} + \{ \mathcal{R}^e \}_m + 1 = \{ 0 \} \]  

(3.6)

where

\[ \Delta \bar{U}^e \equiv (\bar{U}^e)^{m+1} - (\bar{U}^e)^m, \quad m - \text{time level} \]

\[ [M^e] = \int_{\Omega^e} [\Psi]^T [\Psi] dV \]  

(3.7)

\[ \{ \mathcal{R}^e \} = -\int_{\Omega^e} [\nabla \Psi]^T \cdot \{ F^e \} dV + \int_{S^e} [\Psi]^T \{ F_n \} dS \]  

(3.8)

Details of the expression \( \{ \mathcal{R}^e \} \) in equation (3.8) are given in Appendix II. In the standard finite element algorithms, the element equations (3.6) are linearized, usually by Newton method, and all the element equations are assembled to derive a global system of linear equations which are solved by sparse matrix solvers. For large scale problems the matrices involved become too big to be practical. Here we develop a matrix-free relaxation method to solve the non-linear equations directly by a modified Gauss-Seidel iteration.

### 3.2 Locally Implicit Scheme

We wish to solve the non-linear finite element equations iteratively at a node \( i \). We assume the nodal values of the solution at all the surrounding nodes from their latest iterates. The test function \( \Psi_i \), corresponding to the node \( i \), in equation (3.6) gives the contribution of element \( \Omega^e \) to the node \( i \) in the finite element approximation. Adding similar equations from all the elements surrounding a node \( ND \) yields the nodal finite element equation. Thus the equations corresponding to a single node, \( ND \) are

\[ \sum_e \left( \frac{1}{\Delta t} [M^e] \{ \Delta U^e \} + \{ \mathcal{R}^e \}_m + 1 \right)_{ND} = 0 \]  

(3.9)

where \( \Delta \bar{U}^e \) is replaced by \( U^e \) for convenience. Thus \( U^e \) is the conservation variable vector at all the nodes of the element \( e \), and the summation in equation (3.9) is over all elements \( e \) surrounding the node \( ND \). Equation (3.9) represents 5 equations at \( ND \) corresponding to each of the 5 conservation equations. The \( \alpha \)th conservation equation at \( ND \) can be written as

\[ \left[ \sum_e \frac{1}{\Delta t} \int_{\Omega^e} \left( \sum_{j=1}^9 \Delta U_{\alpha,j} \Psi_j \right) \Psi_{(ND)}^e dV - \int_{\Omega^e} \nabla \Psi_{(ND)}^e \cdot F_{\alpha}^{(m+1)} dV \right. \\
+ \left. \int_{\partial \Omega^e} \Psi_{(ND)}^e \cdot F_{\alpha}^{(m+1)} \cdot \vec{n} dS \right] = 0 \]  

(3.10)
where $\Psi^{e}_{(ND)} = \Psi^{e}_{i}$ with $i$ corresponding to the local index of the global node $ND$ in element $e$. For all interior nodes $ND$, the surface flux integral in equation (3.10) vanishes. This equation couples $U$ at all the nodes surrounding the node $ND$. We develop a modified symmetric non-linear Gauss-Seidel iteration to solve the coupled system of non-linear equations directly without linearization. This leads to a matrix-free algorithm for the solution.

For a particular time step $n$, the iteration is carried out as follows. During the iteration process, we assume that all $U$'s in the $\alpha$th equation other than $U_{\alpha}$ are known from the previous step of the iteration. We solve for $\Delta U_{\alpha}$ at node $ND$ approximately by a modified Gauss-Seidel iteration.

\begin{equation}
\Delta U_{\alpha,j}^{(m+1)} = \Delta U_{\alpha,j}^{(m)} + dU_{\alpha,j} \tag{3.11}
\end{equation}

for all nodes $j$ where $(m + 1)$th iterates are not available.

\begin{equation}
\tilde{F}_{\alpha}^{(n+1)} (n+1) \simeq \tilde{F}_{\alpha}^{(n+1)} \left( U^{n} + \Delta U^{(m+1)} \right) \tag{3.12}
\end{equation}

at nodes where $\Delta U^{(m+1)}$ is available. At other nodes where only $\Delta U^{(m)}$ is available,

\begin{equation}
\tilde{F}_{\alpha}^{(n+1)} \simeq \tilde{F}_{\alpha} \left( U^{n} + \Delta U^{(m)} + dU \right) \tag{3.13}
\end{equation}

\begin{equation}
\simeq \tilde{F}_{\alpha} \left( U^{n} + \Delta U^{(m)} \right) + \frac{\partial \tilde{F}}{\partial U} dU
\end{equation}

The Jacobian matrices $\frac{\partial \tilde{F}}{\partial U}$ have inviscid and viscous parts $\frac{\partial \tilde{F}_{\text{Invis}}}{\partial U}$, $\frac{\partial \tilde{F}_{\text{Vis}}}{\partial U}$ respectively. The inviscid part is approximated by the spectral radii of the Jacobian matrices multiplied by identity matrices.

\begin{equation}
\frac{\partial \tilde{F}_{\text{Invis}}}{\partial U} \rightarrow (|u| + a, |v| + a, |w| + a) I = \tilde{S} \tilde{R} \tag{3.14}
\end{equation}

where $u, v, w$ are velocity components and $a$ is the speed of sound. The viscous parts of the Jacobian matrices are not altered. For the iterative corrections $dU$'s we make the approximation,

\begin{equation}
dU_{\alpha,j} \simeq dU_{\alpha,(ND)} \tag{3.15}
\end{equation}

for all the nodes $j$ at which the latest iterates are not available. $dU_{\alpha,(ND)} = dU_{\alpha,i}$ where $i$ is the local index corresponding to the global node $ND$. With this approximation, we obtain explicit scalar expression for the iterative correction at the node $ND$, $dU_{\alpha,(ND)}$.

\begin{equation}
C dU_{\alpha,ND} = -Re d\zeta_{\alpha,ND} \tag{3.16}
\end{equation}
The superscript (*) corresponds to the iteration level \((m)\) or \((m+1)\) which ever is available at the nodes surrounding the node \((ND)\).

\[
C = \sum_{e} \left[ \frac{1}{\Delta t} \int_{\Omega^e} \sum_{j} \Psi_j^e \Psi_{(ND)}^{IND(j)} dV \right] + \sum_{e} \int_{\Omega^e} |\vec{\nabla} \Psi_{(ND)}^e| \cdot S \mathcal{R} \Psi_{(ND)}^e dV + \sum_{e} \left[ \int_{\Omega^e} \vec{\nabla} \Psi_{(ND)}^e \cdot \sum_{j} IND(j) \frac{\partial \vec{F}^a V_{is}}{\partial U_{\alpha,j}} dV \right]
\]

\[
IND(j) = \begin{cases} 
1 & \text{for nodes } j \text{ at iteration level } m \\
0 & \text{for nodes } j \text{ at iteration level } m + 1
\end{cases}
\]

The absolute value sign \(\cdot\) in the middle integral indicates the absolute values of each of its components. In defining the coefficient \(C\), contributions of surface integrals do not exist for all interior nodes and they are ignored for boundary nodes for simplicity. Approximations made in \(C\) to simplify the algorithm while preserving numerical stability for large Courant numbers, do not affect the solution which is obtained by driving \(Res\) function to zero. One iteration sweep starting from the initial node to the final node followed by a reverse sweep makes one symmetric sweep. Typically two symmetric sweeps per time step are sufficient for obtaining time asymptotic solutions.

### 3.3 Surface Flux Computation

Volume integrals over quadrilateral brick elements are computed by isoparametric transformations to a standard cube and by the use of two point Gaussian integration in each direction. The details of such computations are available in many books on finite element methods. Surface flux computation, however, is less known and the basic idea is outlined below.

Suppose \(\xi, \eta, \zeta\) are the local coordinates and \(x, y, z\) are global coordinates and we wish to compute the surface flux on the surface \(\zeta = 1\) of an element.
\[ Q (x(\xi + \Delta \xi, \eta, s), y(\xi + \Delta \xi, \eta, s), z(\xi + \Delta \xi, \eta, s)) \]

\[ O(x,y,z) \]

\[ P (x(\xi, \eta + \Delta \eta, s), y(\xi, \eta + \Delta \eta, s), z(\xi, \eta + \Delta \eta, s)) \]

\[ \int_{\zeta=1} \vec{F} \cdot \vec{\tau} dS = \int_{\zeta=1} \vec{F} \cdot d\vec{S} \]  
(3.20)

\[ d\vec{S} = \vec{\tau} dS = OP \times OQ \]

\[ = (x \Delta \xi, y \Delta \xi, z \Delta \xi) \times (x \Delta \eta, y \Delta \eta, z \Delta \eta) \]  
(3.21)

\[ = \left( \frac{\partial(y,z)}{\partial(\xi,\eta)}, \frac{\partial(z,x)}{\partial(\xi,\eta)}, \frac{\partial(x,y)}{\partial(\xi,\eta)} \right) d\xi \, d\eta \]

\[ \int_{\zeta=1} \vec{F} \cdot d\vec{S} \] can now be computed with Gaussian integration in \( \xi \) and \( \eta \) directions, at \( \zeta = 1 \). The values of \( \vec{F} \) and the surface Jacobians are evaluated at the Gaussian points on the surfaces of the elements, in contrast to the interior evaluation of volume integral computations.

### 3.4 Artificial Dissipation

Though the scheme is linearly stable, non-linear numerical instabilities could arise in strongly convective flows. Various artificial dissipation terms have been developed in the literature to suppress the numerical instabilities. The features we seek for artificial dissipation terms are that they only suppress numerical instabilities, they be smaller than the real viscous terms, they are of higher order than the truncation terms and finally they should be implementable in the code without excessive computation. For this purpose, we choose the adaptive artificial dissipation terms of third order similar to those developed by Jameson\(^{(1)}\) and others. These terms are included in the finite element code. A listing of the code is given in Appendix III.
4. Test Calculations

4.1 Couette Flow

The first test problem is the simulation of a one dimensional shear flow under pressure gradient. It has been computed with a uniform mesh of $2 \times 6 \times 2$ linear (eight-node) elements with the following boundary conditions.

$$
u = v = w = 0 \text{ at } y = 0 \text{ plane}$$
$$u = U_0, \quad v = w = 0 \text{ at } y = 6 \text{ plane}$$
$$w = 0 \text{ at } z = 0 \text{ and } z = 2 \text{ plane}$$
$$v = 0 \text{ at } z = 0 \text{ and } x = 2 \text{ plane}$$

A favorable pressure gradient of $\frac{\partial p}{\partial x} = -1$ is imposed. Fig. 4 shows the computed solution with wall velocity $U_0 = 3$. This problem has a simple exact solution as given in Schlichting$^6$. The computed solution agrees with the exact solution and the two are indistinguishable on the plot. For this simple problem, it takes very few time steps to reach a steady state solution starting from uniform flow conditions. The table of global and local correspondence of nodes, typical of finite element codes is also shown in Fig. 4.

4.2 Laminar Boundary Layer Over a Flat Plate

As another check case, laminar boundary layer over a flat plate has been computed with a stretched mesh of $4 \times 6 \times 1$ linear elements. In this problem the convective terms are of the same order as some of the viscous terms. The finite element solution for a Reynolds number of $Re = 10^4$, along with the boundary conditions and the mesh used are shown in Fig. 5. The computed solution agrees qualitatively with the exact solution even with a very coarse mesh. A converged solution can also be obtained for $Re = 10^5$.

4.3 Flow Over an Airfoil

The locally implicit scheme for two dimensional Navier–Stokes equations with finite volume discretization is applied to compute airfoil flows. Calculations have been carried out with the code and comparisons have been made with experimental results. High Reynolds number viscous flows over an RAE 2822 airfoil have been computed from subsonic to transonic Mach numbers. An algebraic turbulence model developed by Baldwin and Lomax$^5$ has been incorporated into the code. A body conforming C-grid (128 x 32) for an RAE 2822 airfoil is shown in Fig. 6. The mesh spacing normal to the airfoil is highly stretched to resolve turbulent viscous layer. The spacing ranges from $0.00005$ to $3$ chord lengths from inner to outer grid lines. Mach contours for turbulent flow at Mach number, $M = 0.6$, angle of attack, $\alpha = 2.57$ and Reynolds number, $Re = 6.3 \times 10^6$ are shown in Fig. 7a. Fig. 7b shows the corresponding $C_p$ plot where numerical results are compared.
with experimental values published by Cook, McDonald and Firmin\textsuperscript{(8)}. The agreement of numerical and experimental values for $C_p$ is reasonable for a relatively coarse grid. Similar Mach contour and $C_p$ plots are presented for transonic flow case with $M = 0.725$, $\alpha = 2.92$ and $Re = 6.5 \times 10^6$ in Figs. 8a and 8b.

4.4 Flow in a Turn-around Duct

As a test for the 3-D finite element code, flow in an axisymmetric turnaround duct is computed at Mach number = 0.1. The schematic sketch of the turnaround duct is shown in Fig. 9. The geometry used corresponds to a test rig at Rockwell International which is shown in Fig. 10. A relatively coarse grid of $8 \times 15 \times 2$ elements are chosen. Since the flow is axisymmetric, 3 sectional planes with 2 elements in the circumferential direction are chosen and flow is set to be the same in each of the planes in the boundary conditions. The grid in one of the constant-angle planes and the computed velocity vectors are shown in Fig. 11 and a more detailed view of the velocity vectors in the bend region are shown in Fig. 12. The flow features are qualitatively correct. But a finer grid computation is necessary for quantitative comparisons with experimental results and it will be carried out later.
5. References


Fig. 1 Amplification Factor for Different Courant Numbers ($\bar{R} = R = C/64$)
Fig. 2 Amplification Factor for Different Dissipation Parameters ($C = 10$, $\bar{R} = R$)
Fig. 3 Amplification Factor for Different Dissipation Parameters ($C = 10$, $\bar{R} = 2R$)
Correspondence of nodes:

<table>
<thead>
<tr>
<th>Local</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
</tr>
</tbody>
</table>

Fig. 4 Couette Flow
Fig. 5 Flat Plate Boundary Layer Flow
Fig. 6  Computational Grid for Viscous Flows
RAE 2822 Airfoil – C grid (128 x 32)
Fig. 7a Mach Number Contours for Viscous Flow
RAE 2822 Airfoil - $M_\infty = 0.6$, $\alpha = 2.57^\circ$, $Re = 6.3 \times 10^6$
Fig. 7b Numerical and Experimental Pressure Coefficients
RAE 2822 Airfoil – $M_{\infty} = 0.6, \ \alpha = 2.57^\circ, \ Re = 6.3 \times 10^6$
Fig. 8a  Mach Number Contours for Viscous Flow
RAE 2822 Airfoil – $M_\infty = 0.725, \, \alpha = 2.92^\circ, \, Re = 6.5 \times 10^6$
Fig. 8b Numerical and Experimental Pressure Coefficients
RAE 2822 Airfoil – $M_\infty = 0.725$, $\alpha = 2.92^\circ$, $Re = 6.5 \times 10^6$
Fig. 9 Sketch of a Section of a Turnaround Duct
Fig. 10 Geometry of a Test Rig for a Turnaround Duct

HOT-WIRE-PROBE PORTS LOCATED TO ALLOW
DETECTION OF FLOW CHANGES THROUGH BEND

END VIEW OF
TURNAROUND BASE

SECTION AA
SHOWING TYPICAL DUAL-PROBE HOT-WIRE
ORIENTATION FOR $\alpha = 12.8^\circ$
TRAVISER LINES FOR OTHER LOCATIONS SHOWN.
Fig. 11 Computational Grid and Velocity Vectors in a Cross Section of the Turnaround Duct
Fig. 12 Velocity Vectors in the Re Bend Region of the Turnaround Duct
APPENDIX I

The details of the Unsteady Compressible Navier-Stokes equations, which are used in the finite element code are given below. The equations are written in conservation form as

\[
\frac{\partial U}{\partial t} + \nabla \cdot \{\vec{F}^v\} + \nabla \cdot \{\vec{F}^I\} = \{0\}
\]

where

\[
U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho \varepsilon \end{pmatrix}, \quad \vec{F}^v = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho \varepsilon \end{pmatrix}, \quad \vec{F}^I = \begin{pmatrix} \rho \vec{v} \\ \rho \vec{v} \cdot \vec{v} + pI \\ \vec{v} (\rho \varepsilon + p) \end{pmatrix}
\]

\[
q = -k \vec{V} T, \quad \tau_{ij} = -\frac{2}{3} \mu \delta_{ij} e_{kk} + 2\mu \epsilon_{ij}
\]

\[
p = (\gamma - 1) \left[ \rho \varepsilon - \frac{p}{2} (u^2 + v^2 + w^2) \right], \quad \epsilon_{ij} = \frac{1}{2} (u_{i,j} + v_{j,i})
\]

The viscous and inviscid fluxes are given by

\[
\vec{F}^v = \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix}_{(5 \times 3)}, \quad \vec{F}^I = \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho vu & \rho v^2 + p & \rho vw \\ \rho uw & \rho vw & \rho w^2 + p \\ u(\rho \varepsilon + p) & v(\rho \varepsilon + p) & w(\rho \varepsilon + p) \end{bmatrix}_{(5 \times 3)}
\]

\[
p = (\gamma - 1) \left[ \varepsilon - \frac{p}{2}(u^2 + v^2 + w^2) \right], \quad (p = \rho RT), \quad e = \rho \varepsilon
\]

- Sutherland's theory of viscosity:

\[
\mu = \mu_0 \left( \frac{T}{T_0} \right)^{\frac{1}{2}} \left( \frac{T_0 + S_1}{T + S_1} \right)
\]

\[S_1 = \text{constant} \ (= 110 \ ^oK \ \text{for air})\]

- Properties of air at 20 °C (= T_0) and atmospheric pressure (p_1 = 1 atm)
\( \rho_0 = 1.205 \text{kg/m}^3 \)
\( p_0 = 0.101325 \times 10^6 \text{N/m}^2 \)
\( T_0 = 20 \degree C = 293 \degree K \)

\[
R = \left( \frac{p_0}{\rho_0 T_0} \right) = 287 \left( \frac{N \cdot m}{Kg \cdot K} \text{ or } \frac{m^2}{Sec^2 - \degree K} \right) 
\]

\( \mu_0 = 17.9 \times 10^{-6} (\text{Pa} - \text{Sec}) \)
\( k = 2.5 \times 10^{-2} (\text{W/m} - \degree K) \)
\( P_r = 0.72 \)
\( \alpha = 0.208 \)
\( \gamma = 1.402 \)

AUXILIARY RELATIONS

\( p = \text{Pressure (N/m}^2) \)
\( T = \text{Temperature (\degree K)} \)
\( \gamma = \frac{C_p}{C_v} \)
\( C_p = \text{Specific heat at constant pressure} \)
\( C_v = \text{Specific heat at constant volume} \)
\( R = \text{Gas constant (N \cdot m/Kg - \degree K)} \)
\( k = \text{Thermal conductivity (W/m - \degree K)} \)
\( \mu_0 = \text{Reference viscosity (Pa - Sec.)} \)
\( T_0 = \text{Reference temperature (\degree K)} \)
\( \rho_0 = \text{Reference density (Kg/m}^3) \)
\( p = \rho RT \)
\( C_p = \frac{\gamma R}{\gamma - 1} \)
\( C_v = \frac{R}{\gamma - 1} \)

\( \alpha = \text{Thermal diffusitivity, } = \frac{k}{\rho C_p} \)

\( P_r = \text{Prandtl number } = \frac{\mu C_p}{k} \)

\( M_{\infty} = \text{Mach number } = \frac{U_{\infty}}{C_{\infty}} \)
APPENDIX II
Details of Finite Element Equations

The details of finite element equations which approximate the Navier-Stokes equations are given below. In equation (3.8) the residual \( \{ R^e \} \) has two parts. One is a volume integral, \( R_v \), and the other is a surface integral, \( R_s \).

\[
\{ R^e \} = \{ R_v \} + \{ R_s \}
\]

where

\[
\{ R_v \} = -\int_{\Omega^*} [\nabla \Psi]^T \{ \vec{F} \} dV
\]

\[
\{ R_s \} = \int_{\delta \Omega^*} [\Psi]^T \{ F_n \} dS
\]

The components of \( \{ R_v \} \) for \( \Psi \) which corresponds to a node \( I \) are given by

\[
R^1_v = -\int_{\Omega^*} \left( \frac{\partial \Psi_I}{\partial x} U_2 + \frac{\partial \Psi_I}{\partial y} U_3 + \frac{\partial \Psi_I}{\partial z} U_4 \right) dV
\]

\[
R^2_v = -\int_{\Omega^*} \left\{ \left( \frac{U_2^2}{U_1} + p \right) \frac{\partial \Psi_I}{\partial x} + \frac{U_2 U_3}{U_1} \frac{\partial \Psi_I}{\partial y} + \frac{U_2 U_4}{U_1} \frac{\partial \Psi_I}{\partial z}
\right.
\]

\[
\left. + \frac{\partial \Psi_I}{\partial x} \left[ \frac{2}{3} \mu \left( -2 \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) + \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right) \right]
\]

\[
\left. + \frac{\partial \Psi_I}{\partial y} \left[ -\mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) \right) \right]
\]

\[
\left. + \frac{\partial \Psi_I}{\partial z} \left[ -\mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) \right) \right] \right\} dV
\]

where

\[
\frac{\partial}{\partial x_i} \left( \frac{U_\alpha}{U_1} \right) = \frac{1}{U_1} \left( \frac{\partial U_\alpha}{\partial x_i} - \frac{U_\alpha \partial U_1}{U_1 \partial x_i} \right)
\]

\[
R^3_v = -\int_{\Omega^*} \left\{ \left( \frac{\partial \Psi_I}{\partial x} \cdot \frac{U_2 U_3}{U_1} + \frac{U_3^2}{U_1} + p \right) \frac{\partial \Psi_I}{\partial y} + \frac{U_3 U_4}{U_1} \frac{\partial \Psi_I}{\partial z}
\right.
\]

\[
\left. + \frac{\partial \Psi_I}{\partial x} \left[ -\mu \frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) - \mu \frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) \right]
\]

\[
\left. + \frac{\partial \Psi_I}{\partial y} \left[ \frac{2}{3} \mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) - 2 \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) \right) \right]
\]

\[
\left. + \frac{\partial \Psi_I}{\partial z} \left[ -\mu \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) - \mu \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) \right] \right\} dV
\]
\[ R^4_v = - \int_{\Omega^*} \left\{ \frac{\partial \Psi_I}{\partial x} \frac{U_2 U_4}{U_1} + \frac{\partial \Psi_I}{\partial y} \frac{U_3 U_4}{U_1} + \left( \frac{U_2^2}{U_1} + p \right) \frac{\partial \Psi_I}{\partial z} + \frac{\partial \Psi_I}{\partial x} \left[ - \mu \frac{\partial}{\partial z} \left( \frac{U_3}{U_1} \right) - \mu \frac{\partial}{\partial x} \left( \frac{U_4}{U_1} \right) \right] + \frac{\partial \Psi_I}{\partial y} \left[ - \mu \frac{\partial}{\partial z} \left( \frac{U_2}{U_1} \right) - \mu \frac{\partial}{\partial y} \left( \frac{U_4}{U_1} \right) \right] + \frac{\partial \Psi_I}{\partial z} \left[ 2 \mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) - 2 \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right) \right] \right\} dV \]

\[ R^5_v = - \int_{\Omega^*} \left\{ \frac{U_2}{U_1} \frac{U_5 + p}{U_1} \frac{\partial \Psi_I}{\partial x} + \frac{U_3}{U_1} \frac{U_5 + p}{U_1} \frac{\partial \Psi_I}{\partial y} + \frac{U_4}{U_1} \frac{U_5 + p}{U_1} \frac{\partial \Psi_I}{\partial z} - \frac{2}{3} \mu \frac{U_2}{U_1} \frac{\partial \Psi_I}{\partial x} \left[ 2 \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) - \frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) - \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right] \right. \]

\[ - \mu \frac{U_1}{U_1} \frac{\partial \Psi_I}{\partial y} \left[ \frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) \right] \]

\[ - \mu \frac{U_1}{U_1} \frac{\partial \Psi_I}{\partial z} \left[ \frac{\partial}{\partial z} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) \right] \]

\[ - \mu \frac{U_1}{U_1} \frac{\partial \Psi_I}{\partial y} \left[ \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) + \frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) \right] \]

\[ - \mu \frac{U_1}{U_1} \frac{\partial \Psi_I}{\partial z} \left[ \frac{\partial}{\partial z} \left( \frac{U_3}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) \right] \]

\[ - \mu \frac{U_1}{U_1} \frac{\partial \Psi_I}{\partial y} \left[ 2 \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) - \frac{\partial}{\partial x} \left( \frac{U_4}{U_1} \right) - \frac{\partial}{\partial y} \left( \frac{U_4}{U_1} \right) \right] \]

\[ - k \left[ \frac{\partial \Psi_I \partial Q}{\partial x} \frac{\partial \Psi_I \partial Q}{\partial y} + \frac{\partial \Psi_I \partial Q}{\partial z} \frac{\partial \Psi_I \partial Q}{\partial z} \right] \}

\text{d}V \]

where

\[ Q = \frac{1}{U_1} \left[ U_5 - \frac{1}{2U_1} (U_2^2 + U_3^2 + U_4^2) \right] \]

For defining the components of \( \{ R_s \} \) we write

\[ F_n dS = \vec{F} \cdot \vec{n} dS = \vec{F} \cdot d\vec{S} \]

\[ = \vec{F} \cdot \left( \frac{\partial (y, z)}{\partial (\xi, \eta)}, \frac{\partial (z, x)}{\partial (\xi, \eta)}, \frac{\partial (x, y)}{\partial (\xi, \eta)} \right) d\xi d\eta \]

\[ - 34 - \]
as derived in equation (11) of the last report\(^{(3)}\), for a typical surface, say \(\zeta = 1\) of an element.

Denote

\[
(V_1, V_2, V_3) = \left( \frac{\partial(y, z)}{\partial(x, y)}, \frac{\partial(z, x)}{\partial(x, y)}, \frac{\partial(x, y)}{\partial(x, y)} \right)
\]

Now the components of \(\{R_s\}\) for \(\Psi_I\) which corresponds to a node \(I\), for a typical surface \(\zeta = 1\) of an element can be written as

\[
R_s^1 = \int_{\Omega} (V_1 U_2 + V_2 U_3 + V_3 U_4) \Psi_I d\xi \, d\eta
\]

\[
R_s^2 = \int_{\Omega} \left\{ \left( \frac{U_2^2}{U_1} + p \right) V_1 + \frac{U_3 U_2}{U_1} V_2 + \frac{U_3 U_4}{U_1} V_3 
+ V_1 \left[ \frac{2}{3} \mu \left( \frac{2}{\partial U_2} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) + \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right) \right]
+ V_2 \left[ -\mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) \right) \right]
+ V_3 \left[ -\mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial z} \left( \frac{U_2}{U_1} \right) \right) \right] \right\} \Psi_I d\xi \, d\eta
\]

where

\[
\frac{\partial}{\partial x_i} \left( \frac{U_a}{U_1} \right) = \frac{1}{U_1} \left( \frac{\partial U_a}{\partial x_i} - \frac{U_a}{U_1} \frac{\partial U_1}{\partial x_i} \right)
\]

\[
R_s^3 = \int_{\Omega} \left\{ \frac{U_2 U_3}{U_1} V_1 + \left( \frac{U_2^2}{U_1} + p \right) V_2 + \frac{U_3 U_4}{U_1} V_3 
+ V_1 \left[ -\frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) - \frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) \right]
+ V_2 \left[ \frac{2}{3} \mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) - \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right) \right]
+ V_3 \left[ -\frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) - \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) \right] \right\} \Psi_I d\xi \, d\eta
\]

\[
R_s^4 = \int_{\Omega} \left\{ \frac{U_2 U_4}{U_1} V_1 + \frac{U_3 U_4}{U_1} V_2 + \left( \frac{U_4^2}{U_1} + p \right) V_3 
+ V_1 \left[ -\frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) - \frac{\partial}{\partial z} \left( \frac{U_2}{U_1} \right) \right]
+ V_2 \left[ -\frac{\partial}{\partial x} \left( \frac{U_4}{U_1} \right) - \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right]
+ V_3 \left[ \frac{2}{3} \mu \left( \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) - \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right) \right] \right\} \Psi_I d\xi \, d\eta
\]
\[ R_s^j = \int_{\partial \Omega^e} \left\{ \frac{U_2}{U_1}(U_5 + p)V_1 + \frac{U_3}{U_1}(U_5 + p)V_2 + \frac{U_4}{U_1}(U_5 + p)V_3 ight. \\
\left. - \frac{2}{3} \mu \frac{U_2}{U_1} V_1 \left[ \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) - \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) - \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right] \\
- \mu \frac{U_3}{U_1} V_1 \left[ \frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) \right] \\
- \mu \frac{U_4}{U_1} V_1 \left[ \frac{\partial}{\partial z} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial x} \left( \frac{U_4}{U_1} \right) \right] \\
- \mu \frac{U_2}{U_1} V_2 \left[ \frac{\partial}{\partial y} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) \right] \\
\left. - \frac{2}{3} \mu \frac{U_3}{U_1} V_2 \left[ \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) - \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) - \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) \right] \\
- \mu \frac{U_4}{U_1} V_2 \left[ \frac{\partial}{\partial z} \left( \frac{U_3}{U_1} \right) + \frac{\partial}{\partial y} \left( \frac{U_4}{U_1} \right) \right] \\
- \mu \frac{U_2}{U_1} V_3 \left[ \frac{\partial}{\partial z} \left( \frac{U_2}{U_1} \right) + \frac{\partial}{\partial x} \left( \frac{U_3}{U_1} \right) \right] \\
- \mu \frac{U_3}{U_1} V_3 \left[ \frac{\partial}{\partial z} \left( \frac{U_3}{U_1} \right) + \frac{\partial}{\partial x} \left( \frac{U_4}{U_1} \right) \right] \\
- \frac{2}{3} \mu \frac{U_4}{U_1} V_3 \left[ \frac{\partial}{\partial z} \left( \frac{U_4}{U_1} \right) - \frac{\partial}{\partial x} \left( \frac{U_2}{U_1} \right) - \frac{\partial}{\partial y} \left( \frac{U_3}{U_1} \right) \right] \\
\left. - k \left[ \frac{\partial \Psi_i}{\partial x} \frac{\partial Q}{\partial x} + \frac{\partial \Psi_i}{\partial y} \frac{\partial Q}{\partial y} + \frac{\partial \Psi_i}{\partial z} \frac{\partial Q}{\partial z} \right] \right\} \Psi_j \, d\xi \, d\eta \]

where

\[ Q = \frac{1}{U_1} \left[ U_5 - \frac{1}{2U_1}(U_2^3 + U_3^2 + U_4^2) \right] \]

Components of \( \{ R_s \} \) for other surfaces of an element can be written similarly.

The coefficient \( C \) of equation (3.13) has volume integrals of the derivatives of viscous flux terms. The details of those integrals are given below.

Denote

\[ \int_{\Omega^e} \nabla \Psi_{(ND)}^e \cdot \frac{\partial F_{\alpha}^{\text{Vis}}}{\partial U_{\alpha,i}} \, dV = N_{(ND),i}^e \]

Subscript \((ND)\) corresponds to the local index \(i\) of the global node \(ND\) in element \(e\). These integrals can be written as

\[ N_{ij}^1 = 0 \]

\[ N_{ij}^2 = \mu \int_{\Omega^e} \left[ \frac{4}{3} \frac{\partial \Psi_i}{\partial z} \frac{\partial}{\partial x} \left( \frac{\Psi_j}{U_1} \right) + \frac{\partial \Psi_i}{\partial y} \frac{\partial}{\partial y} \left( \frac{\Psi_j}{U_1} \right) + \frac{\partial \Psi_i}{\partial z} \frac{\partial}{\partial z} \left( \frac{\Psi_j}{U_1} \right) \right] \, dV \]

where

\[ \frac{\partial}{\partial x} \left( \frac{\Psi_j}{U_1} \right) = \frac{1}{U_1} \left[ \frac{\partial \Psi_j}{\partial x} - \Psi_j \frac{\partial U_1}{\partial x} \right], \text{ etc.,} \]
\[ N_{ij}^3 = \mu \int_{\Omega^*} \left[ \frac{\partial \Psi_i}{\partial x} \frac{\partial}{\partial x} \left( \frac{\psi_j}{U_1} \right) + \frac{4}{3} \frac{\partial \Psi_i}{\partial y} \frac{\partial}{\partial y} \left( \frac{\psi_j}{U_1} \right) + \frac{\partial \Psi_i}{\partial z} \frac{\partial}{\partial z} \left( \frac{\psi_j}{U_1} \right) \right] dV \]

\[ N_{ij}^4 = \mu \int_{\Omega^*} \left[ \frac{\partial \Psi_i}{\partial x} \frac{\partial}{\partial x} \left( \frac{\psi_j}{U_1} \right) + \frac{4}{3} \frac{\partial \Psi_i}{\partial y} \frac{\partial}{\partial y} \left( \frac{\psi_j}{U_1} \right) + \frac{\partial \Psi_i}{\partial z} \frac{\partial}{\partial z} \left( \frac{\psi_j}{U_1} \right) \right] dV \]

\[ N_{ij}^5 = \hat{k} \int_{\Omega^*} \left[ \frac{\partial \Psi_i}{\partial x} \frac{\partial}{\partial x} \left( \frac{\psi_j}{U_1} \right) + \frac{4}{3} \frac{\partial \Psi_i}{\partial y} \frac{\partial}{\partial y} \left( \frac{\psi_j}{U_1} \right) + \frac{\partial \Psi_i}{\partial z} \frac{\partial}{\partial z} \left( \frac{\psi_j}{U_1} \right) \right] dV \]
APPENDIX III
**Finite-Element Analysis of Flows of Viscous, Compressible Fluids in Three-Dimensional Enclosures.**

This program is developed by Professors J. N. Reddy of Virginia Polytechnic Institute and K. C. Reddy of the University of Tennessee Space Institute. The program is under continuous development during April '86 to present. Unauthorized use of the program is prohibited.

Developed: April 1986 - Present

### Description of the Variables

**CFL**
- The Courant-Friedrichs-Levy number

**ELXYZ**
- Array of element coordinates of nodes

**IBNDC**
- Array of boundary nodes for different variables

**IORDER**
- Order of the equations to be solved

**ISTART**
- Restart index (1=Restart; 0=New Start)

**KELSUR**
- A two-dimensional array that contains element number and local number of its surface that requires flux computation:
  - **KELSUR(I,1)** = Global element number of the global I-th surface
  - **KELSUR(I,2)** = Local surface number of the global I-th surface

**KNDSUR**
- A two-dimensional (M by 4) array which contains global surface numbers surrounding a node that requires flux computation. Here M denotes the number of nodes requiring flux computation:
  - **KNDSUR(I,J)** = Global number of the local J-th surface associated with the I-th boundary node that requires flux computation.

**MEN**
- Maximum number of elements at a node

**MNE**
- Maximum number of nodes per element

**NDF**
- No. of unknowns at each node

**NDSURF**
- Array containing the sequential number of the boundary nodes which require flux computation or containing zero:
  - **NDSURF(I)=0**, if no surfaces around the I-th node requires flux computation.
  - **NDSURF(I)=J**, if the I-th node requires flux computation; here J denotes the sequential number of node I in the list of surfaces that require flux computation.

**NELEM**
- Connectivity matrix relating global node to
ELEMENTS AROUND THE NODE:

NELEM(I,M)=GLOBAL ELEMENT NUMBER CORRESPONDING TO THE M-TH LOCAL ELEMENT SURROUNDING GLOBAL NODE I (MAXIMUM VALUE OF M IS 8).

NEM........NUMBER OF ELEMENTS IN THE MESH
NGP........NUMBER OF GAUSSIAN POINTS

NMSH......INDICATOR FOR GENERATING MESH:

NMSH=0, MESH INFORMATION IS TO BE READ
NMSH>0, MESH IS GENERATED BY THE PROGRAM (ONLY FOR PRISMATIC AND TAD DOMAINS)

NNM.......NUMBER OF NODES IN THE MESH

NODES....BOOLEAN MATRIX RELATING LOCAL NODES TO GLOBAL NODES OF ELEMENTS:

NODES(N,J)=GLOBAL NODE NUMBER CORRESPONDING TO THE J-TH LOCAL NODE OF ELEMENT N.

NSURF.....TOTAL NUMBER OF SURFACES THAT REQUIRE FLUX COMPUTATION
NTMSTP....NO. OF TIME STEPS

U.........ARRAY OF FIVE PRIMARY Unknowns:
RHO, RHO*U, RHO*V, RHO*W, RHO*E

X,Y,Z.....GLOBAL COORDINATES OF THE NODES

SUBROUTINES USED

BCUPDT....UPDATES THE BOUNDARY CONDITIONS AT THE END OF EACH ITERATION OR TIME STEP.
BNDRY.....Generates array 'KNDSUR', containing surfaces requiring flux computation.
COEFNIT....Generates the coefficient values of each variable at each node of the mesh.
DISPTN....Computes the dissipation model.
DSFSUR....Computes the derivatives of the shape functions at Gauss points of a surface.
FLUXES....Computes flux for each variable at each node of the mesh.
GCSURF....Generates array 'GC', which contains the derivative of X(I) W.R.T. XI(J).
GMETRY....Generates arrays 'SF', 'CNST', 'GDSF' and 'VOL' globally.
INITIAL....Generates initial conditions on boundary faces.
INVDET....Computes the inverse of the Jacobian matrix.
MATMUL....Computes the product of two matrices.
SHAPEL ... evaluates the shape functions and their derivatives at the Gauss points.

SURFGM ... computes components of the unit normal at Gauss points of each boundary surface.

TADMESH ... generates the mesh (X, Y and Z coordinates and array 'NODES') for the Turn-Around-Duct (TAD).

IMPLICIT REAL*8 (A-H,O-Z)
PARAMETER (NNM-432, NEM-240, MXE-8, NGP-2, NDIM-3, NPE-8, NDF-5,
1 NBS=600)
DIMENSION X(NNM), Y(NNM), Z(NNM), TITLE(20), UOLD(NNM, 6), U(NNM, 6),
2 NODIAND(NEM, NPE), NELEM(NNM, MXE), EXYZ(NPE, NDIM), EO(NNM),
3 JORDER(NPE), DIS4(NNM, 6), DC4(NNM), DELU(NPE, 6), AMU(NNM),
4 GDSF(MXE, NPE, NGP, NGP, NGP, NDIM), GNORM(NDIM, NBS, NGP, NGP),
5 SF(NPE, NGP, NGP, NGP), CNST(MXE, NGP, NGP, NGP), EMU(NPE),
6 VOLN(NNM), VOL(MXE), DSURF(NDIM, NPE, 6, NGP, NGP),
7 ELU(NPE, 6), IEL(MXE), IBNDC(NNM, NDF), MINDX(NPE),
8 KELSUR(NBS, 2), KNSUR(NBS, 4), NDSURF(NNM)
COMMON/GMT/SN22(8,8), SN33(8,8), SN44(8,8), SN55(8,8)
COMMON/DTA/GAMA, AMU0, TEMPO, S1, R0, GPR, GAM1, CFL
DATA IORDER/1, 2, 3, 4, 5/
DATA IN, IT/5, 6/

READ(5,2000) TITLE
READ(5,*) ISTART, NMSH, ITER, NTMSTP, CFL, RLXOUT, RLXIN
READ(5,*) AMU0, TEMPO, S1, R0, GAM1, AMACH, DST0
IF(NMSH.EQ.0) GOTO 5

CALL TADMESH(X, Y, Z, IBNDC, KELSUR, NODES, NSURF, NNM, NBS, NDF, NEM, NPE)

END OF THE INPUT DATA

OPEN THE OUTPUT FILE IN WHICH THE DATA IS TO BE STORED.
THE NAME OF THE FILE IS 'TEST' AND THE DATA IS STORED IN THE FORM
OF BINARY NUMBERS.

10 CONTINUE
IREC=30000
OPEN(UNIT=08, FILE='TEST', STATUS='NEW', ACCESS='DIRECT',
  FORM='UNFORMATTED', RECL=IREC, ACTION='READWRITE')
IF(ISTART.EQ.1) THEN
  OPEN(UNIT=07, FILE='RSTART', STATUS='OLD', ACCESS='DIRECT',
    FORM='UNFORMATTED', RECL=IREC, ACTION='READWRITE')
ENDIF

GENERATE ARRAY 'NELEM' USING ARRAY 'NODES'

DO 40 I=1,NNM
DO 15 L=1,MXE
15 NELEM(I,L)=0
ICNT=0
DO 30 J=1,NEM
DO 20 K=1,8
JK=NODES(J,K)
IF (I.NEJK) GOTO 20
ICNT=ICNT+1
NELEM(I,ICNT)=J
IF (ICNT.EQ.MXE) GOTO 40
GOTO 30
20 CONTINUE
30 CONTINUE
40 CONTINUE

DEFINE FIXED PARAMETERS

NGP T=NGP*NGP*NGP
GAM1=GAMA-1.0
GPR=GAMA/PR

INITIALIZE THE FLOW FIELD

NINIT=0
IF (ISTART .EQ. 0) THEN
    CALL INITIAL(NDF,NNM,AMACH,AMU0,TEMP0,S1,R0,GAMA,PR,U,DNST0)
    CALL BCUPDT(NNM,GAMA,R0,TEMP0,U,DNST0)
ELSE
    READ (07,REC=I) NINIT,U
END IF
NTMSTP = NTMSTP + NINIT
NINIT=NINIT+1
DO 50 II=1,6
DO 50 JJ=I,NNM
UOLD(JJ,II)=U(JJ,II)
50 CONTINUE

WRITE OUT INPUT DATA

WRITE(IT,2600)
WRITE(IT,2500)
WRITE(IT,2600)
WRITE(IT,3000) TITLE
WRITE(IT,2100) AMU0,TEMP0,S1,R0,GAMA,PR,DNST0
WRITE(IT,2200) ITER,NTMSTP,CFL,RLXOUT,RLXIN
WRITE(IT,741) AMACH
741 FORMAT(10X,'FREE STREAM MACH NUMBER =',El0.4)
WRITE(IT,3500)
DO 70 I = 1, NEM
WRITE(IT,4000) I,(NODES(I,J),J=1,8)
WRITE(IT,4500)
DO 80 I = 1, NNM
WRITE(IT,4000) I,(NELEM(I,J),J=1,MXE)
WRITE(IT,5500)
DO 90 I = 1, NNM
WRITE(IT,5000) I,X(I),Y(I),Z(I)
WRITE(IT,6100)
DO 100 I=1,NNM
FIND MAX. NO. OF NODES PER EACH ELEMENT, COMPUTE ELEMENTAL
VOLUMES, SHAPE FUNCTIONS AND THEIR GLOBAL DERIVATIVES, AND
THE PRODUCT OF THE WEIGHTS AND THE DETERMINANT OF THE JACOBIAN
MATRIX FOR EACH GAUSS POINT OF EACH ELEMENT.

DO 155 ND=1,NNM

COMPUTE THE NUMBER OF ELEMENTS AROUND NODE 'ND'

DO 115 J=I,MXE
IF (NELEM(ND,J).EQ.0) GOTO 120

CONTINUE
J=MXE+1
120 NUMEL=J-1

initialize the arrays
VOLND(ND)=0.0
DC4(ND)=7*NUMEL

compute array 'IEL' which contains local node corr to node ND

DO 150 N=I,NUMEL
NEL=NELEM(ND,N)
DO 140 I=I,NPE
NI=NODES(NEL,I)
IF (NI.EQ.ND) IEL(N)=I
ELXYZ(I,1)=X(NI)
ELXYZ(I,2)=Y(NI)
ELXYZ(I,3)=Z(NI)

CALL GMETRY(NNM,NEM,MXE,N,NPE,NGP,ELXYZ,GF,SDF,CNST,VOL,
1,NDIM,IEL(N))

CALL BNDRY(NBS,NEM,NNM,NPE,NSURF,NODES,KELS,KDS,NDSURF,KNDSUR)
CALL DSFSUR(DSURF,NPS,PS,NDIM)
* WRITE(8000) (V(SURF(I)),I=1,NEM)
* WRITE(IT,4000) ((KELSUR(I,J),J=1,2),I=1,NSURF)
* WRITE(IT,4000) (KDSURF(I),I=1,16)
* WRITE(IT,4000) ((KNSURS(I,J),J=1,4),I=1,NSURF)

DO 180 NDS=1,NSURF
KE=KELSUR(NDS,1)
K1=KELSUR(NDS,2)
DO 160 I=1,NPE
NI=NODES(KE,I)
ELXYZ(I,1)=X(NI)
ELXYZ(I,2)=Y(NI)
160 ELXYZ(I,3)=Z(NI)

CALL SURFGM(K1,NDS,ELXYZ,DSURF,GNORM,NBS,NPS,PS,NDIM)
BEGIN THE DO-LOOP ON THE NUMBER OF TIME STEPS TO COMPUTE THE SOLUTION

ERROR = 0.0
DO 800 ITMSTP = NINIT, NTMSTP
WRITE (IT, 6000) ITMSTP
DO 190 I = 1, NN

190 AMU (I) = AMU0 * (((TEMP / TEMP0) ** 1.5) * ((TEMP + SI) / (TEMP + S1))

CALL SUBROUTINE 'DISPTN' TO COMPUTE GLOBAL ARTIFICIAL DISSIPATION

CALL DISPTN (NNM, NEM, MXE, X, Y, Z, U, DC4, NODES, NELEM, DIS4, NPE, E0, VOLND)

SYMMETRIC NONLINEAR GAUSS-SEIDEL ITERATION LOOP BEGINS HERE

ITMAX = 2 * ITER
DO 700 ITR = IT, ITMAX
IF (MOD (ITR, 2) .EQ. 1) THEN
    NBEGIN = I
    NEND = NNM
    NINC = 1
ELSE
    NBEGIN = NNM
    NEND = I
    NINC = -1
ENDIF
WRITE (IT, 4007) ITR, ITMAX
BEGIN THE DO-LOOP ON THE NUMBER OF NODES TO COMPUTE THE SOLUTION

DO 600 ND = NBEGIN, NEND, NINC
WRITE (IT, 4006) NBEGIN, NEND, NINC, ND

COMPUTE THE NUMBER OF ELEMENTS (NUMEL) SURROUNDING A NODE

READ (08, REC=ND) ID, CNST, GDSF, VOL, NUMEL, IEL, SN22, SN33, 1 SN44, SN55
IF (ID .NE. ND) THEN
    PRINT *, 'ERROR IN THE READ OF FILES'
STOP
ENDIF

NSTART = 1
NLAST = 5
INCR = 1
DO 500 LOOP = 1, 1

DO-LOOP ON THE NUMBER OF CONSERVATION EQUATIONS BEGINS HERE

DO 400 NEQ = NSTART, NLAST, INCR
WRITE (IT, 4004) NSTART, NLAST, INCR, NEQ, LOOP

LEQ = IORDER (NEQ)
IF (IBNDC (ND, LEQ) .EQ. 0) GOTO 400

DO-LOOP ON THE NUMBER OF ELEMENTS SURROUNDING NODE 'ND' BEGINS HERE
GCM=0.0
GCKVIS=0.0
GCKINV=0.0
TCOEF=0.0
TRES=0.0
TFLX=0.0
DO 300 N=1,NUMEL
WRITE(IT, 4003) NUMEL, N
NEL=NELEM(ND,N)

TRANSFER GLOBAL INFORMATION TO ELEMENT 'NEL'

DO 260 I=1,NPE
MINDX(I)=0
NI=NODES(NEL,I)
EMU(I)=AMU(NI)
IF (NINC.EQ.1 .AND. NI.GE.ND) MINDX(I)=1
IF (NINC.EQ.-1 .AND. NI.LE.ND) MINDX(I)=1
DO 260 II=1,6
DELU(I,II)=U(NI,II)-UOLD(NI,II)
260 ELU(I,II)=U(NI,II)

CALL SUBROUTINE 'COEFNT' TO COMPUTE THE COEFFICIENTS FOR THE EQN

CALL COEFNT (IEL(N), LEQ, N, NPE, NEM, NGP, ELU, SF, GDSF, CNST, VOL, RES, CM, EMU, DELU, MINDX, CKINV, NDF, NDIM, NGPT, MXE)

GOTO (271,272,273,274,275), LEQ

271 GCKVIS=0.0
GOTO 276
272 DO 282 J1=1,NPE
282 GCKVIS=GCKVIS+SN22(N,J1)*MINDX(J1)
GOTO 276
273 DO 283 J1=1,NPE
283 GCKVIS=GCKVIS+SN33(N,J1)*MINDX(J1)
GOTO 276
274 DO 284 J1=1,NPE
284 GCKVIS=GCKVIS+SN44(N,J1)*MINDX(J1)
GOTO 276
275 DO 285 J1=1,NPE
285 GCKVIS=GCKVIS+SN55(N,J1)*MINDX(J1)
276 CONTINUE
GCM=GCM+CM
GCKINV=GCKINV+CKINV
TRES=TRES+RES
GCKINV=GCKINV*8.0/NUMEL
GCKVIS=GCKVIS*AMU(ND)/U(ND,1)
IF (LEQ.EQ.5) GCKVIS=GCKVIS*GPR
TCOEF=GCM+DABS(GCKINV)+GCKVIS
TCOEF=TCOEF+DC4(ND)
IF (NDSURF(ND).EQ.0) GOTO 340
DO 335 J=1,4
KG1=KINDSUR(NDSURF(ND),J)
IF (KG1.EQ.0) GOTO 340
KI=KELSUR(KG1,2)
KL=KELSUR(KG1,1)
DO 310 II=1,NPE
IF (NELEM(ND,II).EQ.KL) THEN
NI=II
GOTO 315
ENDIF
310 CONTINUE
315 DO 330 II=1,NPE
EMU(II)=AMU(NODES(KL,II))
DO 320 J1=1,NDF
C
C
C
CALL FLUXES (LI, LEQ, NI, NPE, NGP, ELU, SF, GDSF, GNORM, KL, KGI, FLX, IEMU, MXE, NBS, NDF, NDIM)
C
TFLX=TFLX+FLX
CONTINUE
IF (LEQ.NE.2) GOTO 350
ERROR0 = ERROR
ERROR = DMAX1 (ERROR0, DABS (TRES + TFLX))
IF (ERROR .GT. ERROR0) MAXND = ND
C
DIS4 (ND, LEQ) = 0.0
DU = -(TRES + TFLX - DIS4 (ND, LEQ)) / TCOEF
U (ND, LEQ) = U (ND, LEQ) + DU * RLXINU (ND, 6) = GAM1 * (U (ND, 5) - 0.5 * (U (ND, 2) * U (ND, 2) + U (ND, 3) * U (ND, 3) + U (ND, 4) * U (ND, 4)) / U (ND, 1))
WRITE (IT, 7500) LEQ, ND, TRES, TFLX, TCOEF, U (ND, LEQ)
400 CONTINUE
NTEMP = NSTART
NSTART = NLAST
NLAST = NTEM-
INCR = -1 * INCR
CONTINUE
WRITE (6, 9999) ND, (U (ND, LI), LI = 1, 6)
*9999 FORMAT (I5, 6E15.7)
600 CONTINUE

END OF THE COMPUTATION FOR ALL NODES IN THE SWEEP

NTEMP = NBEGIN
NBEGIN = NEND
NEND = NTEMP
NINC = -1 * NINC

RESET THE VALUES AT INFLOW, OUTFLOW AND RADIAL SYMMETRY PLANES

---

CALL BCUPDT (NNM, GAM1, R0, TEMPO, U, DNST0)
---

700 CONTINUE

RELAXATION OF THE UPDATED SOLUTION AND COMPUTATION OF PRESSURE

DO 720 II = 1, 5
DO 720 JJ = 1, NNM
U (JJ, II) = UOLD (JJ, II) + RLXOUT * (U (JJ, II) - UOLD (JJ, II))
720 UOLD (JJ, II) = U (JJ, II)
DO 730 JI = 1, NNM
U (JI, 6) = GAM1 * (U (JI, 5) - 0.5 * (U (JI, 2) * U (JI, 2) + U (JI, 3) * U (JI, 3) + U (JI, 4) * U (JI, 4)) / U (JI, 1))
730 UOLD (JI, 6) = U (JI, 6)
WRITE (IT, 7000) ERROR, MAXND
DO 750 I = 1, NNM
750 WRITE (IT, 6500) I, (U (I, J), J = 1, 6)
800 CONTINUE

OPEN (UNIT=09, FILE=' ROLD', STATUS='NEW', ACCESS='DIRECT', * FORM='UNFORMATTED', RECL=IREC, ACTION='READWRITE')
WRITE (09, REC=1) NTEMS, U
STOP

FORMATS
I000 FORMAT (5X,'ARRAYS: KELSUR, NDSURF AND KNDSUR:',/)  
2000 FORMAT (20A4)  
2100 FORMAT (/2X,'PROBLEM DATA:',/)  
2  
/5X,'REFERENCE VISCOSITY (AMU0) ............ =',E12.4,  
3  
/5X,'REFERENCE TEMPERATURE (TEMP0) ........ =',E12.4,  
4  
/5X,'SUTHERLANDS CONSTANT (S1) ............ =',E12.4,  
5  
/5X,'GAS CONSTANT (R0) ..................... =',E12.4,  
6  
/5X,'RATIO OF SPECIFIC HEATS (GAMA) ........ =',E12.4,  
7  
/5X,'PRANDTL NUMBER (PR) ................... =',E12.4,  
8  
/5X,'REFERENCE DENSITY (DNST0) ............. =',E12.4,/)  
2200 FORMAT (/2X,'PARAMETERS OF APPROX.:',/)  
2  
/5X,'NUMBER OF ITERATIONS PER TIME STEP .... =',I3,  
3  
/5X,'NUMBER OF TIME STEPS (NTMSTP) ........ =',I3,  
4  
/5X,'THE CFL NUMBER (CFL) .............. =',E12.4,  
5  
/5X,'OUTER RELAXATION PARAMETER (RLXOUT) ... =',E12.4,  
6  
/5X,'INNER RELAXATION PARAMETER (RLXIN) .... =',E12.4,/)  
2500 FORMAT (/15X,'OUTPUT FROM PROGRAM COMPR3D:',/)  
2600 FORMAT (80(' ')  
3000 FORMAT (1H1,20A4)  
3500 FORMAT (/2X,'CONNECTIVITY MATRIX:',/,  
* 2X, (ELEMENT-TO-NODES)',/)  
4000 FORMAT (I5,2X,I11S)  
4002 FORMAT (5X,'DO-LOOP 200 :',/,915)  
4003 FORMAT (5X,'DO-LOOP 300 :',/,915)  
4004 FORMAT (5X,'DO-LOOP 400 :',/,915)  
4005 FORMAT (5X,'DO-LOOP 500 :',/,915)  
4006 FORMAT (5X,'DO-LOOP 600 :',/,915)  
4007 FORMAT (5X,'DO-LOOP 700 :',/,915)  
4008 FORMAT (5X,'DO-LOOP 800 :',/,915)  
4500 FORMAT (/2X,'CONNECTIVITY ARRAY :',/,  
* 2X, (NODE-TO-ELEMENTS)',/)  
5000 FORMAT (I5,3(2X,E12.4))  
5500 FORMAT (/2X,(X,Y,Z)-COORDINATES OF NODES:',/)  
6000 FORMAT (/2X,'TIME STEP =',I5,/)  
6100 FORMAT (/2X,'INITIAL FIELD VALUES:',/)  
6200 FORMAT (/2X,'SPECIFIED NODAL QUANTITIES (=0, SPECIFIED):',/)  
6300 FORMAT (/2X,'ELEMENT NUMBERS AND THEIR SURFACES THAT REQUIRE FLUX  
* COMPUTATION:',/)  
6500 FORMAT (I5,6E12.4)  
7000 FORMAT (/5X,'MAX. ERROR =',E12.4,/,5X,'NODE NUMBER =',I5,/)  
7500 FORMAT (/5X,'LEQ =',I2,2X,'NODE =',I4,2X,'RESIDUAL =',E12.4,2X,  
* 'FLUX =',E12.4,2X,'TCOF =',E12.4,2X,'SOLN. =',E12.4/)  
8000 FORMAT (5X,'VOLUME OF EACH ELEMENT :',/,5X,6E12.4)  
END  

SUBROUTINE BCUPDT (NNM, GAMA, R0, TEMP0, U, DNST0)  
IMPLICIT REAL*8 (A-H,O-Z)  
COMMON/MSH/ARCANG, NX, NY, NZ, NX1, NX2, NX3  
DIMENSION U(NNM, 6)  

DEFINE FIXED PARAMETERS  
ANX=0.0  
ANY=DSIN(0.5*ARCANG)  
ANZ=DCOS(0.5*ARCANG)  
GAMA=GAMA-1.0  
NXX=NX+1  
NY=NY+1  
NZ=NZ+1  

SET THE NORMAL VELOCITY TO ZERO AT THE MIDPLANE  
DO 30 IX=1,NXX
DO 30 IY=1,NYY
ND=(IX-1)*NYY*NZZ+NYY+IY
U (ND,3)=U (ND,3) * (1.0-ANY*ANY) -U (ND,4) *ANY*ANZ
U (ND,4)=-U (ND,3) *ANY*ANZ+U (ND,4) * (1.0-ANZ*ANZ)
U (ND,5)=U (ND,6) /GAMI+0.5*(U (ND,2) *U (ND,2) +U (ND,3) *U (ND,3) +
* U (ND,4) *U (ND,4)) /U (ND,1)
RESET THE VALUES ON PARALLEL PLANES TO THOSE ON THE MIDPLANE

ND1=ND-NYY
ND2=ND+NYY
U (ND1,1)=U (ND,1)
U (ND1,2)=U (ND,2)
U (ND1,3)=U (ND,3) *ANZ-U (ND,4) *ANY
U (ND1,4)=U (ND,3) *ANY+U (ND,4) *ANZ
U (ND1,5)=U (ND,5)
U (ND1,6)=U (ND,6)
U (ND2,1)=U (ND,1)
U (ND2,2)=U (ND,2)
U (ND2,3)=U (ND,3) *ANZ+U (ND,4) *ANY
U (ND2,4)=-U (ND,3) *ANY+U (ND,4) *ANZ
U (ND2,5)=U (ND,5)
U (ND2,6)=U (ND,6)
30 CONTINUE

RESET THE VALUES AT OUTFLOW BOUNDARY
DO 40 IZ=1,NZZ
DO 40 IY=1,NYY
ND = IY + (IZ-1) *NYY + NX*NYY*NZZ
U (ND,6)=DNST0*R0*TEMP0*0.98
U (ND,5)=U (ND,6) /GAMI+0.5*(U (ND,2) *U (ND,2) +U (ND,3) *U (ND,3) +
* U (ND,4) *U (ND,4)) /U (ND,1)
40 CONTINUE

SET CONSTANT TEMPERATURE ON THE WALLS
DO 60 KD = I, NX-1
ND1 = (NYY*NZZ)*KD + 1
DO 50 JZ = I, NZZ
ND = ND1 + (JZ-1) *NYY
U (ND,6)=U (ND,5) *GAMI
U (ND,1)=U (ND,6) /R0*TEMP0
50 CONTINUE

NN = ND + NY
U (NN,6)=U (NN,5) *GAMI
U (NN,1)=U (NN,6) /R0*TEMP0
60 CONTINUE
RETURN
END

SUBROUTINE BNDRY(NBS, NEM, NNM, NPE, NSURF, NODES, KELSUR, NDSURF, KNDSUR)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION NODES (NEM, NPE), KELSUR (NBS, 2), KNDSUR (NBS, 4), NDSURF (NNM),
* K(4)
NCOUNT=0
DO 10 I=1,NNM
10 NDSURF (I)=0
DO 20 I=1,4
DO 20 J=1,NSURF
20 KNDSUR (J,I)=0
DO 150 I=1,NSURF
   KEL=KELSUR(I,1)
   KSRF=KELSUR(I,2)
   GOTO(30,40,50,60,70,80),KSRF
30  K(1)=NODES(KEL,1)
    K(2)=NODES(KEL,4)
    K(3)=NODES(KEL,8)
    K(4)=NODES(KEL,5)
    GOTO 90
40  K(1)=NODES(KEL,2)
    K(2)=NODES(KEL,3)
    K(3)=NODES(KEL,7)
    K(4)=NODES(KEL,6)
    GOTO 90
50  K(1)=NODES(KEL,1)
    K(2)=NODES(KEL,5)
    K(3)=NODES(KEL,6)
    K(4)=NODES(KEL,2)
    GOTO 90
60  K(1)=NODES(KEL,4)
    K(2)=NODES(KEL,8)
    K(3)=NODES(KEL,7)
    K(4)=NODES(KEL,3)
    GOTO 90
70  K(1)=NODES(KEL,1)
    K(2)=NODES(KEL,2)
    K(3)=NODES(KEL,3)
    K(4)=NODES(KEL,4)
    GOTO 90
80  K(1)=NODES(KEL,5)
    K(2)=NODES(KEL,6)
    K(3)=NODES(KEL,7)
    K(4)=NODES(KEL,8)
    CONTINUE
90  DO 120 J=1,4
    IF(NDSURF(K(J)).EQ.0)THEN
       NCOUNT=NCOUNT+1
       NDSURF(K(J))=NCOUNT
       KNDSUR(NCOUNT,1)=I
    ELSE
       NC=NDSURF(K(J))
       DO 100 JJ=2,4
           IF(KNDSUR(NC, JJ).EQ.0)THEN
              KNDSUR(NC, JJ)=I
           GOTO 110
    ENDIF
100  CONTINUE
110  CONTINUE
120  CONTINUE
150  CONTINUE
RETURN
END

SUBROUTINE COEFNT (IEL, LEQ, N, NPE, NEM, NGP, ELU, SF, GDSF, CNST, VOL, RES,
                    CM, EMU, DELU, MINDX, CKINV, NDF, NDIM, NGPT, MXE)
------------------------------------------------------------------------
ELU(I,J) ..... ELEMENT SOLUTION VECTOR (J-TH COMPO. AT I-TH NODE)
SF(I,...) ..... SHAPE FUNCTION ASSOCIATED WITH THE I-TH NODE
GDSF(N,J,...) LOCAL DERIVATIVE OF J-TH SHAPE FUNCTION WITH RESPECT TO X(I) COORDINATE
------------------------------------------------------------------------
THIS IS A VECTORIZED VERSION OF THE SUBROUTINE COEFNT

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION SF(NPE,NGP,NGP,NGP), CNST(MXE,NGP,NGP,NGP), VOL(MXE),
2 GDSF(MXE,NPE,NGP,NGP,NGP,NDIM), ELU(NPE,6), EMU(NPE),
3 U(6,8), DU(7,3,8), DUL(7,3,8), U1(6,8), DELU(NPE,6),
4 III(8), JJJ(8), KKK(8), F(8,8), DF(9,9,3), MINDX(NPE),
5 DQI(3), C(8), GMU(8)
COMMON/DTA/GAMA, AMU0, TEMPO, S1, R0, GPR, GAM1, CFL

DATA III/I,2,1,2,1,2,1,2/
DATA JJJ/I,1,2,1,2,1,2,1,2/
DATA KKK/I,1,1,1,2,2,2,2/

CM=0.0
CK=0.0
CKINV=0.0
DLNGTH=0.0
RES=0.0
FMAS=0.0
SPEED=DSQRT(ELU(IEL,6)*GAMA/ELU(IEL,1))

DO 10 L=I,NGPT
C(L) = CNST(N, III(L), JJJ(L), KKK(L))
DO 10 I=NPE
F(L,I) = SF(I, III(L), JJJ(L), KKK(L))

DF(L,1,1) = GDSF(N,I,III(L),JJJ(L),KKK(L),1)
DF(L,1,2) = GDSF(N,I,III(L),JJJ(L),KKK(L),2)
DF(L,1,3) = GDSF(N,I,III(L),JJJ(L),KKK(L),3)

TSPEED=SPEED+(DABS(ELU(IEL,2))+DABS(ELU(IEL,3))+DABS(ELU(IEL,4)))/
* ELU(IEL,1)
DT=CFL*(VOL(N)**(1./3.))/TSPEED

DO 40 J=I,NDF
DO 40 L=I,NGPT
SUM1=0.0
SUM2=0.0
SUM3=0.0
SUM4=0.0
DO 30 I=NPE
SUMI=SUMI+DF(L,I,1)*ELU(I,J)
SUM2=SUM2+DF(L,I,2)*ELU(I,J)
SUM3=SUM3+DF(L,I,3)*ELU(I,J)
30 SUM4=SUM4+F(L,I)*ELU(I,J)
DU(J,1,L)=SUM1
DU(J,2,L)=SUM2
DU(J,3,L)=SUM3
40 U(J,L)=SUM4
DO 50 J=2,4
DO 50 L=1,NGPT
U1(J,L)=U(J,L)/U(1,L)
DU1(J,1,L)=(DU(J,1,L)-U1(J,L)*DU1(1,1,L))
DU1(J,2,L)=(DU(J,2,L)-U1(J,L)*DU1(1,2,L))
50 DU1(J,3,L)=(DU(J,3,L)-U1(J,L)*DU1(1,3,L))

DO 70 J1=1,NPE
DO 60 L=1,NGPT
PROD=F(L,IEL)*F(L,J1)*C(L)
CM=CM+PROD*MINDX(J1)
60 FMAS=FMAS+PROD*DELU(J1,LEQ)
70 CONTINUE

COMPUTE MASS MATRIX TIMES DELU TERM

DO 70 J1=1,NPE
DO 60 L=1,NGPT
PROD=F(L,IEL)*F(L,J1)*C(L)
CM=CM+PROD*MINDX(J1)
60 FMAS=FMAS+PROD*DELU(J1,LEQ)
70 CONTINUE
**COMPUTE INVISCID COEFFICIENT FOR INNER ITERATION**

```c
DO 90 L=1,NGPT
   CKINV=CKINV+ (DABS (DF (L, IEL, 1) * (DABS (UI (2, L)) +SPEED))
    + DABS (DF (L, IEL, 2) * (DABS (UI (3, L)) +SPEED))
    + DABS (DF (L, IEL, 3) * (DABS (UI (4, L)) +SPEED)))*C(L)
   *F(L, IEL)
90 CONTINUE
```

**COMPUTE RESIDUES ETC FOR A CONSERVATION EQUATION**

```c
GOTO(100,200,300,400,500),LEQ

100 DO ii0 L=1,NGPT
   RES=RES- (DF (L, IEL, 1) *U(2, L) +DF (L, IEL, 2) *U(3, L)
            +DF (L, IEL, 3) *U(4, L))*C(L)
110 CONTINUE
GOTO 600

200 DO 240 L=1,NGPT
   SUM=0.0
   DO 220 I=1,NPE
      SUM=SUM+EMU (I) *F (L, I)
   220 GMU (L)=SUM
   DO 260 L=1,NGPT
      U22=U (2, L) *U (2, L)
      U23=U (2, L) *U (3, L)
      U24=U (2, L) *U (4, L)
      U33=U (3, L) *U (3, L)
      U34=U (3, L) *U (4, L)
      U44=U (4, L) *U (4, L)
      PRES=GAM1* (U (5, L) -0.5*(U22+U33+U44)/U(1, L))
      AMU23=2.0*GMU(L)/3.0
      AMU43=2.0*AMU23
      RES=RES-C (L) *((U22+PRES*U(1, L)+AMU23*(-2.0*DU1 (2, 1, L)
                    +DU1 (3, 2, L)+DU1 (4, 3, L)))*DF (L, IEL, 1)
            +U23-GMU (L) *(DU1 (3, 1, L)+DU1 (2, 2, L)))*DF (L, IEL, 2)
            +U24-GMU (L) *(DU1 (4, 1, L)+DU1 (2, 3, L)))*DF (L, IEL, 3))/U (1, L)
260 CONTINUE
GOTO 600

300 DO 340 L=1,NGPT
   SUM=0.0
   DO 320 I=1,NPE
      SUM=SUM+EMU (I) *F (L, I)
   320 GMU (L) =SUM
   DO 360 L=1,NGPT
      U22=U (2, L) *U (2, L)
      U23=U (2, L) *U (3, L)
      U33=U (3, L) *U (3, L)
      U34=U (3, L) *U (4, L)
      U44=U (4, L) *U (4, L)
      PRES=GAM1* (U (5, L) -0.5*(U22+U33+U44)/U(1, L))
      AMU23=2.0*GMU(L)/3.0
      AMU43=2.0*AMU23
      RES=RES-C (L) *((U33+PRES*U(1, L)+AMU23*(-2.0*DU1 (3, 2, L)
                    +DU1 (4, 3, L)+DU1 (2, 1, L)))*DF (L, IEL, 2)
            +(U34-GMU (L) *(DU1 (4, 2, L)+DU1 (3, 3, L)))*DF (L, IEL, 3)
            +(U23-GMU (L) *(DU1 (2, 3, L)+DU1 (3, 1, L)))*DF (L, IEL, 1))/U (1, L)
360 CONTINUE
GOTO 600

400 DO 440 L=1,NGPT
   SUM=0.0
   DO 420 I=1,NPE
      SUM=SUM+EMU (I) *F (L, I)
   420 GMU (L)=SUM
```

```c
```
DO 460 L=1,NGPT
  U22=U(2,L)*U(2,L)
  U24=U(2,L)*U(4,L)
  U33=U(3,L)*U(3,L)
  U34=U(3,L)*U(4,L)
  U44=U(4,L)*U(4,L)
  PRES=GAM1*(U(5,L)-0.5*(U22+U33+U44)/U(1,L))
  AMU23=2.0*GMU(L)/3.0
  AMU43=2.0*AMU23
  RES=RES-C(L)*(U44+PRES*U(1,L)+AMU23*(-2.0*DU1(4,3,L)
  1 +DU1(2,1,L)+DU1(3,2,L))*DF(L,IEL,3)
  2 +(U24-GMU(L)*(DU1(2,3,L)+DU1(4,1,L)))*DF(L,IEL,1)
  3 +(U34-GMU(L)*(DU1(3,3,L)+DU1(4,2,L)))*DF(L,IEL,2))/U(1,L)
  460 CONTINUE
GOTO 600

C
DO 540 L=1,NGPT
  SUM=0.0
DO 520 I=1,NPE
  SUM=SUM+EMU(I)*F(L,I)
  GMU(L)=SUM
DO 560 L=1,NGPT
  U22=U(2,L)*U(2,L)
  U33=U(3,L)*U(3,L)
  U44=U(4,L)*U(4,L)
  PRES=GAM1*(U(5,L)-0.5*(U22+U33+U44)/U(1,L))
  AKH=GMU(L)*GPR
  AMU23=2.0*GMU(L)/3.0
  AMU43=2.0*AMU23
  DQ1(1)=DU(5,1,L)-U1(2,L)*DU(2,1,L)-U1(3,L)*DU(3,1,L)
  2 -U1(4,L)*DU(4,1,L)+DU(1,1,L)*(-U(5,L)/U(1,L)
  3 +U1(2,L)*U1(2,L)+U1(3,L)*U1(3,L)+U1(4,L)*U1(4,L)
  DQ1(2)=DU(5,2,L)-U1(2,L)*DU(2,2,L)-U1(3,L)*DU(3,2,L)
  2 -U1(4,L)*DU(4,2,L)+DU(1,2,L)*(-U(5,L)/U(1,L)
  3 +U1(2,L)*U1(2,L)+U1(3,L)*U1(3,L)+U1(4,L)*U1(4,L)
  DQ1(3)=DU(5,3,L)-U1(2,L)*DU(2,3,L)-U1(3,L)*DU(3,3,L)
  2 -U1(4,L)*DU(4,3,L)+DU(1,3,L)*(-U(5,L)/U(1,L)
  3 +U1(2,L)*U1(2,L)+U1(3,L)*U1(3,L)+U1(4,L)*U1(4,L)
  RES1 = (U(2,L)*U(5,L)+PRES)-AMU23*U1(2,L)*(-2.0*DU1(2,1,L)
  2 -DU1(3,2,L)-DU1(4,3,L))-GMU(L)*U1(3,L)*DU1(2,2,L)
  3 +DU1(3,1,L)))+U1(4,L)*DU1(2,3,L)+DU1(4,1,L))
  4 -AKH*DQ1(1)))*DF(L,IEL,1)
  RES2 = (U(3,L)*U(5,L)+PRES)-AMU23*U1(3,L)*(-2.0*DU1(3,2,L)
  2 -DU1(4,3,L)-DU1(2,1,L)))-GMU(L)*U1(4,L)*DU1(3,3,L)
  3 +DU1(4,2,L)))+U1(2,L)*DU1(3,1,L)+DU1(2,2,L))
  4 -AKH*DQ1(2)))*DF(L,IEL,2)
  RES3 = (U(4,L)*U(5,L)+PRES)-AMU23*U1(4,L)*(-2.0*DU1(4,3,L)
  2 -DU1(2,1,L)-DU1(3,2,L)))-GMU(L)*U1(2,L)*DU1(4,1,L)
  3 +DU1(2,3,L)))+U1(3,L)*DU1(4,2,L)+DU1(3,3,L))
  4 -AKH*DQ1(3)))*DF(L,IEL,3)
  RES = RES - (RES1+RES2+RES3)*C(L)/U(1,L)
  560 CONTINUE
  600 CONTINUE
RES=RES+FMAS/DT
CM=CM/DT
RETURN
END

SUBROUTINE DISPTN(NNM, NEM, MXE, X, Y, Z, U, DC4, NODES, NELEM, DIS4, * 
NPE, EO, VOLND)
IMPLICIT REAL*8 (A-H, O-Z)
C
DATA KAPA2, KAPA4/0.1, 0.01/
DO 50 IE=1,6
DO 40 ND=1,NNM
SUME0=0.0
DO 20 NE=1,MXE
IF (NELEM(ND,NE).EQ.0) GOTO 30
NEL=NELEM(ND,NE)
DO 20 NP=1,NPE
NI=NODES(NEL,NP)
20 SUME0=SUME0+U(NI,IE)-U(ND,IE)
NE=MXE+1
30 CONTINUE
DC4(ND)=7*(NE-1)
40 DIS4(ND,IE)=SUME0
50 CONTINUE
DO 60 ND=1,NNM
DIS4(ND,5)=DIS4(ND,5)+DIS4(ND,6)
60 DIS4(ND,6)=ABS(DIS4(ND,6))/U(ND,6)*KAPA2
COMPUTE THE FOURTH-ORDER DISSIPATION
DO 150 IE=1,5
DO 140 ND=1,NNM
SUMDC=0.0
E0(ND)=0.0
SUMD0=0.0
ISW=1
IF (DIS4(ND,6).GT.KAPA4) ISW=0
DO 120 NE=1,MXE
NEL=NELEM(ND,NE)
IF (NEL.EQ.0) GOTO 130
DO 100 NP=1,NPE
NI=NODES(NEL,NP)
IF (NI.NE.ND) GOTO 100
XL=X(NI)-X(ND)
YL=Y(NI)-Y(ND)
ZL=Z(NI)-Z(ND)
EDGE=DSQRT(XL*XL+YL*YL+ZL*ZL)
EPSLN=(VOLND(ND)+VOLND(NI))*0.5/EDGE
IF (IE.EQ.5) SUMDC=SUMDC+EPSLN*( (DC4(ND)-1.0)*KAPA4*ISW+DIS4(ND,6) )
SUMD0=SUMD0-(DIS4(NI,IE)-DIS4(ND,IE))*EPSLN*KAPA4*ISW
100 CONTINUE
120 CONTINUE
130 CONTINUE
IF (IE.EQ.5) DC4(ND)=SUMDC
140 E0(ND)=SUMD0
DO 150 ND=1,NNM
150 DIS4(ND,IE)=E0(ND)+DIS4(ND,IE)*DIS4(ND,6)
RETURN
END

SUBROUTINE DFSFSUR(DSURF, NGP, NPE, NDIM)
--this subroutine evaluates the derivatives of the shape functions at the gaussian points of the surfaces of an element--
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION XNODE(8,3), XYZ(3), DSURF(NDIM,NPE,6,NGP,NGP), GAUSS(2)
DATA XNODE/-1.0D0, 2*1.0D0,-2*1.0D0, 2*1.0D0,-1.0D0, 2*1.0D0,-1.0D0, 2*1.0D0/
1,2*1.0D0,2*1.0D0,4*1.0D0,4*1.0D0/

\[ FCK(A, B, C) = 0.125*A*B*C \]
\[ SQRT3 = DSQRT(3.0D0) \]
\[ GAUSS(1) = -1.0D0/SQRT3 \]
\[ GAUSS(2) = -GAUSS(1) \]
DO 80 K1 = 1, 6
DO 60 NGP1 = 1, NGP
DO 60 NGP2 = 1, NGP
GOTO (10, 10, 20, 20, 30, 30), K1
10 XYZ(1) = (-1)**K1
XYZ(2) = GAUSS(NGP1)
XYZ(3) = GAUSS(NGP2)
GOTO 40
20 XYZ(2) = (-1)**K1
XYZ(3) = GAUSS(NGP1)
XYZ(1) = GAUSS(NGP2)
GOTO 40
30 XYZ(3) = (-1)**K1
XYZ(1) = GAUSS(NGP1)
XYZ(2) = GAUSS(NGP2)
40 DO 50 I1 = 1, NPE
XNP1 = XYZ(1)*XNODE(I1, 1) + 1.0
YNP1 = XYZ(2)*XNODE(I1, 2) + 1.0
ZNP1 = XYZ(3)*XNODE(I1, 3) + 1.0
DSURF(1, I1, K1, NGP1, NGP2) = FCK(XNODE(I1, 1), YNP1, ZNP1)
DSURF(2, I1, K1, NGP1, NGP2) = FCK(XNP1, XNODE(I1, 2), ZNP1)
DSURF(3, I1, K1, NGP1, NGP2) = FCK(XNP1, YNP1, XNODE(I1, 3))
50 CONTINUE
60 CONTINUE
80 RETURN
END

SUBROUTINE FLUXES (IEL, LEQ, N, NPE, NGP, ELU, SF, GDSF, GNORM, K1, KGI, FLX,
1 EMU, MXE, NBS, NDF, NDIM)

ELU(I, J) ...... ELEMENT SOLUTION VECTOR (J-TH COMPO. AT I-TH NODE)
SF(I, ...) ..... SHAPE FUNCTION ASSOCIATED WITH THE I-TH NODE
GDSF(N, J, I) .. GLOBAL DERIVATIVE OF J-TH SHAPE FUNCTION
WITH RESPECT TO X(I) COORDINATE OF THE N-TH ELEMENT
GDINT(I, J) .... INTERPOLATED GDSF ON SURFACE OF AN ELEMENT
SFINT(I) ...... INTERPOLATED SF ON SURFACE OF AN ELEMENT

IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION SF(NPE, NGP, NGP, NGP), GDSF(MXE, NPE, NGP, NGP, NGP, NDIM),
1 GDINT(8, 3), SFINT(8), GNORM(NDIM, NBS, NGP, NGP), EMU(NPE),
2 ELU(NPE, 6), DU(6, 3), U(6), U1(6), DU1(6, 3), DQ1(3), VECTR(3)
COMMON/DTA/GAMA, AMU0, TEMP0, S1, R0, GPR, GAM1, CFL

K0 = (K1+1)/2
FLX = 0.0
SQRT3 = DSQRT(3.0D0)

DO-LOOP ON GAUSS INTEGRATION BEGINS HERE
DO 200 JJ = 1, NGP
DO 200 KK = 1, NGP
AMU = 0.0
EVALUATE THE COMPONENTS OF THE SURFACE NORMAL AT THE GAUSS POINTS
IF (K0-2) 30, 40, 50

30 NI=1
      NJ=1
      NJ1=NJ
      NK=KK
      NK1=NK
      GOTO 60

40 NJ=1
      NJ1=2
      NK=JJ
      NI=KK
      NJ1=NI
      GOTO 60

50 NK=1
      NK1=2
      NI=JJ
      NJ=KK
      NJ1=NI

C 60 DO 70 I=1, NPE
      FI=SF (I, NI, NJ, NK)
      F2=SF (I, NI1, NJJ, NK1)
      SFINT (I) = ((-1) **K1 * SQRT3 * (F2 - F1) + F2 + F1) / 2.0
      F3=GDGDF (N, I, NI, NJ, NK, 1)
      F4=GDGDF (N, I, NI1, NJJ, NK1, 1)
      GDINT (I, 1) = ((-1) **K1 * SQRT3 * (F4 - F3) + F4 + F3) / 2.0
      F3=GDGDF (N, I, NI, NJ, NK, 2)
      F4=GDGDF (N, I, NI1, NJJ, NK1, 2)
      GDINT (I, 2) = ((-1) **K1 * SQRT3 * (F4 - F3) + F4 + F3) / 2.0
      F3=GDGDF (N, I, NI, NJ, NK, 3)
      F4=GDGDF (N, I, NI1, NJJ, NK1, 3)
      GDINT (I, 3) = ((-1) **K1 * SQRT3 * (F4 - F3) + F4 + F3) / 2.0

70 AMU=AMU+SFINT (I) * EMU (I)

C 80 DO 100 J=1, NDF
     SUM1=0.0
     SUM2=0.0
     SUM3=0.0
     SUM4=0.0
     DO 90 I=1, NPE
     SUMI=SUMI+GDINT (I, I) * ELU (I, J)
     SUM2=SUM2+GDINT (I, 2) * ELU (I, J)
     SUM3=SUM3+GDINT (I, 3) * ELU (I, J)
     SUM4=SUM4+SFINT (I) * ELU (I, J)

90 DU (J, I) = SUM1
      DU (J, 2) = SUM2
      DU (J, 3) = SUM3

100 U (J) = SUM4
     U1 (2) = U (2) / U (1)
     U1 (3) = U (3) / U (1)
     U1 (4) = U (4) / U (1)
     DO 110 J=2, 4
     DU1 (J, 1) = (DU (J, 1) - U1 (J) * DU (1, 1))
     DU1 (J, 2) = (DU (J, 2) - U1 (J) * DU (1, 2))

110 DU1 (J, 3) = (DU (J, 3) - U1 (J) * DU (1, 3))
     VECTR (1) = GNORM (1, KG1, JJ, KK)
     VECTR (2) = GNORM (2, KG1, JJ, KK)
     VECTR (3) = GNORM (3, KG1, JJ, KK)

C COMPUTE PRESSURE, TEMPERATURE, VISCOSITY USING THE SUTHERLAND'S
C LAW, AND THE DIFFUSION CONSTANT AT THE GAUSS POINTS

U22 = U (2) * U (2)
U23 = U (2) * U (3)
U24 = U(2) * U(4)
U33 = U(3) * U(3)
U34 = U(3) * U(4)
U44 = U(4) * U(4)
PRES = GAM1 * (U(5) - 0.5 * (U22 + U33 + U44)) / U(1)

AKH = AMU * GPR
AMU23 = 2.0 * AMU / 3.0
AMU43 = 2.0 * AMU23

COMPUTE THE FLUX FOR EACH CONSERVATION EQUATION AT THE NODE

GOTO (140, 150, 160, 170, 180), LEQ
140 FLX = FLX + (U(2) * VECTR(I) + U(3) * VECTR(2) + U(4) * VECTR(3)) * SFINT(IEL)
GOTO 200

150 FLX = FLX + ((U22 + PRES * U(I) + AMU23 * (-2.0 * DUI(2, I) + DUI(3, 2) + DUI(4, 3)))  
* VECTR(1) + (U23 - AMU * (DUI(3, I) + DUI(2, 3))) * VECTR(2)  
+ (U24 - AMU * (DUI(4, I) + DUI(2, 3))) * VECTR(3)) * SFINT(IEL) / U(I)
GOTO 200

160 FLX = FLX + ((U33 + PRES * U(I) + AMU23 * (-2.0 * DUI(3, 2) + DUI(4, 3) + DUI(2, 1)))  
* VECTR(2) + (U34 - AMU * (DUI(4, 2) + DUI(3, 3))) * VECTR(3)  
+ (U34 - AMU * (DUI(4, 2) + DUI(3, 3))) * VECTR(3)) * SFINT(IEL) / U(I)
GOTO 200

170 FLX = FLX + ((U44 + PRES * U(I) + AMU23 * (-2.0 * DUI(4, 3) + DUI(2, 1) + DUI(3, 2)))  
* VECTR(3) + (U45 - AMU * (DUI(2, 1) + DUI(4, 1))) * VECTR(1)  
+ (U45 - AMU * (DUI(2, 1) + DUI(4, 1))) * VECTR(1)) * SFINT(IEL) / U(I)
GOTO 200

180 DQI(1) = DUI(5, 1) - U1(2) * DUI(2, 1) - U1(3) * DUI(3, 1) - U1(4) * DUI(4, 1)
2   + DUI(1, 1) * (-U(5) / U(1) + U1(2) + U1(3) + U1(4) + U1(4))
2   + DUI(2, 2) * U1(2) + U1(3) + U1(4) + U1(4)
2   + DUI(3, 3) * U1(3) + U1(4) + U1(4)
2   + DUI(4, 4) * U1(4) + U1(4)

FLX = FLX + ((U(2) * U(5) + PRES) - AMU23 * U1(2) * (2.0 * DUI(2, 1) + DUI(3, 2))  
- DUI(4, 3) - AMU * (U1(3) + DUI(2, 1) + DUI(3, 1) + U1(4) + U1(4))  
+ DUI(4, 1)) * AKH * DQI(1) * VECTR(1)

RETURN

SUBROUTINE GCSURF (GC, DSURF, ELXYZ, NGPI, NGPK, NGP, K1, NDIM, NPE)

GC(I, J) ..... DERIVATIVE OF X(I) W.R.T. XI(J)
DSURF(I, J, K) ..... DERIVATIVE OF PSI(J) W.R.T. XI(I), J=1,..., NPE,
ON K-TH SURFACE OF MASTER ELEMENT

IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION ELXYZ(NPE, NDIM), DSURF(NDIM, NPE, 6, NGP, NGP), GC(NDIM, NDIM)

DO 200 I = 1, NDIM
DO 200 K = 1, NDIM
SUM = 0.0
DO 100 J = 1, NPE
100 SUM = SUM + DSURF(K, J, K1, NGPI, NGPK) * ELXYZ(J, I)
200 GC(I, K) = SUM
RETURN
END
SUBROUTINE GMETRY(NNM, NEM, MXE, N, NPE, NGP, ELXYZ, SF, GDSF, CNST, VOL, 
1 NDIM, IEL)

SF(I,II,JJ, KK) .......... I-TH SHAPE FUNCTION AT THE (II,JJ, KK)-TH GAUSS POINT
GDSF(N,I,II,JJ, KK,J) ... GLOBAL DERIVATIVE OF I-TH SHAPE FUNCTION WITH RESPECT TO THE X(J) COORDINATE FOR ELEMENT N
DSF(I,J) .............. LOCAL DERIVATIVE OF I-TH SHAPE FUNCTION WITH RESPECT TO J-TH LOCAL COORDINATE
ELXYZ(I,J) ............ J-TH GLOBAL COORDINATE OF I-TH NODE
XYZ(II) ............... II-TH GAUSSIAN POINT

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION SF(NPE,NGP,NGP,NGP), CNST(MXE,NGP,NGP,NGP), VOL(MXE), 
2 GDSF(MXE,NPE,NGP,NGP,NDIM), ELXYZ(NPE,NDIM), WT(2),
3 GAUSS(2), GJ(3,3), XYZ(3), GJINV(3,3), DSF(3,8), GDSFL(3,8),
4 SFL(8)
COMMON/GMT/SN22(8,8), SN33(8,8), SN44(8,8), SN55(8,8)
DATA NCOUNT/0/

C SQRT3=DSQRT(3.0D0)
GAUSS(1)=-1.0D0/SQRT3
GAUSS(2)=-GAUSS(1)
WT(1)=1.0D0
WT(2)=1.0D0
DO-LOOP ON GAUSS INTEGRATION BEGINS HERE

VOL(N)=0.0
DO 50 J=I,NPE
SN22(N,J)=0.0
SN33(N,J)=0.0
SN44(N,J)=0.0
SN55(N,J)=0.0
DO 200 II=I,NGP
  DO 200 JJ=I,NGP
    DO 200 KK=I,NGP
      XYZ(1)=GAUSS(II)
      XYZ(2)=GAUSS(JJ)
      XYZ(3)=GAUSS(KK)
      CALL SHAPEL(XYZ, SFL, DSF, NDIM, NPE)
      CALL MATMUL(DSF, ELXYZ, GJ, NDIM, NPE, NDIM)
      CALL INVDET(GJ, GJINV, DET)
      CALL MATMUL(GJINV, DSF, GDSFL, NDIM, NDIM, NPE)

CNST(N,II, JJ, KK)=DET*WT(II)*WT(JJ)*WT(KK)
DO 150 I=I,NPE
  SN22(N,I)=SN22(N,I)+(4.0/3.0*GDSFL(1,IEL)*GDSFL(1,I)+
      1 GDSFL(2,IEL)*GDSFL(2,I)+GDSFL(3,IEL)*GDSFL(3,I))*CNST(N,II, JJ, KK)
  SN33(N,I)=SN33(N,I)+(4.0/3.0*GDSFL(2,IEL)*GDSFL(2,I)+
      1 GDSFL(3,IEL)*GDSFL(3,I)+GDSFL(1,IEL)*GDSFL(1,I))*CNST(N,II, JJ, KK)
  SN44(N,I)=SN44(N,I)+(4.0/3.0*GDSFL(3,IEL)*GDSFL(3,I)+
      1 GDSFL(1,IEL)*GDSFL(1,I)+GDSFL(2,IEL)*GDSFL(2,I))*CNST(N,II, JJ, KK)
  SN55(N,I)=SN55(N,I)+(GDSFL(1,IEL)*GDSFL(1,I)+
      1 GDSFL(2,IEL)*GDSFL(2,I)+GDSFL(3,IEL)*GDSFL(3,I))*CNST(N,II, JJ, KK)

C ****************************************************************
C CALL SHAPEL(XYZ, SFL, DSF, NDIM, NPE)
C CALL MATMUL(DSF, ELXYZ, GJ, NDIM, NPE, NDIM)
C CALL INVDET(GJ, GJINV, DET)
C CALL MATMUL(GJINV, DSF, GDSFL, NDIM, NPE)
C ****************************************************************
C CNST(N,II, JJ, KK)=DET*WT(II)*WT(JJ)*WT(KK)
DO 150 I=I,NPE
  SN22(N,I)=SN22(N,I)+(4.0/3.0*GDSFL(1,IEL)*GDSFL(1,I)+
      1 GDSFL(2,IEL)*GDSFL(2,I)+GDSFL(3,IEL)*GDSFL(3,I))*CNST(N,II, JJ, KK)
  SN33(N,I)=SN33(N,I)+(4.0/3.0*GDSFL(2,IEL)*GDSFL(2,I)+
      1 GDSFL(3,IEL)*GDSFL(3,I)+GDSFL(1,IEL)*GDSFL(1,I))*CNST(N,II, JJ, KK)
  SN44(N,I)=SN44(N,I)+(4.0/3.0*GDSFL(3,IEL)*GDSFL(3,I)+
      1 GDSFL(1,IEL)*GDSFL(1,I)+GDSFL(2,IEL)*GDSFL(2,I))*CNST(N,II, JJ, KK)
  SN55(N,I)=SN55(N,I)+(GDSFL(1,IEL)*GDSFL(1,I)+
      1 GDSFL(2,IEL)*GDSFL(2,I)+GDSFL(3,IEL)*GDSFL(3,I))*CNST(N,II, JJ, KK)
SUBROUTINE INTIAL (NDF, NNM, AMACH, AMU0, TEMP0, SI, R0, GAMA, PR, U, DNST0)

INITIAL CONDITIONS FOR THE TURN-AROUND-DUCT PROBLEM
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/MSH/ARCANG, NX, NY, NZ, NXI, NX2, NX3
DIMENSION U(NNM, 6)

DEFINE FIXED PARAMETERS
GAM1=GAMA-1.0
NYY=NY+1
NZZ=NZ+1

INITIALIZE THE FLOW FIELD
DO i0 J=2,4
DO i0 I=1,NNM
10 U(I,J)=0.0
DO 20 IZ=1,NZZ
DO 20 IY=2,NY
ND = IY + (IZ-1)*NYY
U(ND,2)=-DSQRT (GAMA*R0 *TEMP0) *AMACH
IF (IY.EQ.2 .OR. IY.EQ. 8)U (ND, 2) =U (ND, 2) *0.1885
IF (IY.EQ.3.OR. IY.EQ.7) U (ND, 2) =U (ND, 2) *0.5066
IF (IY.EQ.4.OR. IY.EQ.6)U (ND, 2) =U (ND, 2) *0.8393
CONTINUE

INITIALIZE THE MID PLANE
DO 30 IX = 2,NXI+1
DO 30 IY = 2,NY
ND = (IX-1)*NYY*NZZ+NYY+IY
NDI= NYY+IY
30 U(ND,2) = U(NDI,2)
PI = ATAN(1.0)*4.0
DO 40 IX = NXI+2,NX1+NX2
DO 40 IY = 2,NY
ND = (IX-1)*NYY*NZZ+NYY+IY
NDI= NYY+IY
U(ND,2) = U(NDI,2) *COS((IX-NXI-I)*PI/NX2)
U(ND,3) =-U(NDI,2) *SIN((IX-NXI-I)*PI/NX2)
CONTINUE

U(ND, 3) AND U(ND, 4) ARE ZERO (HENCE, U(ND, 5) IS AS DEFINED BELOW)
SUBROUTINE INVDAT(A, B, DET)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(3,3), B(3,3)
G(Z1, Z2, Z3, Z4) = Z1*Z2-Z3*Z4
F(Z1, Z2, Z3, Z4) = G(Z1, Z2, Z3, Z4)/DET
C1 = G(A(2, 2), A(3, 3), A(2, 3), A(3, 2))
C2 = G(A(2, 3), A(3, 1), A(2, 1), A(3, 3))
C3 = G(A(2, 1), A(3, 2), A(2, 2), A(3, 1))
DET = A(1, 1)*C1 + A(1, 2)*C2 + A(1, 3)*C3
B(1, 1) = F(A(2, 2), A(3, 3), A(2, 3), A(3, 2))
B(1, 2) = -F(A(2, 1), A(3, 3), A(2, 1), A(3, 2))
B(1, 3) = -F(A(2, 1), A(2, 3), A(1, 3), A(2, 2))
B(2, 1) = -F(A(2, 1), A(3, 3), A(2, 3), A(3, 1))
B(2, 2) = F(A(1, 1), A(3, 3), A(3, 1), A(1, 3))
B(2, 3) = -F(A(1, 1), A(2, 3), A(1, 3), A(2, 1))
B(3, 1) = F(A(2, 1), A(3, 2), A(3, 1), A(2, 2))
B(3, 2) = -F(A(1, 1), A(3, 2), A(1, 2), A(3, 1))
B(3, 3) = F(A(1, 1), A(2, 2), A(2, 1), A(1, 2))
RETURN
END

SUBROUTINE MATMUL(A, B, C, M, N, L)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(M, N), B(N, L), C(M, L)
DO 20 I=1, M
DO 20 J=1, L
SUM = 0.0
DO 10 K=1, N
SUM = SUM + A(I, K) * B(K, J)
20 C(I, J) = SUM
RETURN
END

SUBROUTINE SHAPEL(XYZ, SF, DF, NDIM, NPE)
SHAPE FUNCTIONS FOR LINEAR, ISOPARAMETRIC 3-DIMENSIONAL ELEMENT
THIS SUBROUTINE EVALUATES THE SHAPE FUNCTIONS AND THEIR FIRST
DERIVATIVES AT THE GAUSSIAN POINT XYZ
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION XNODE(8,3), XYZ(NDIM), SF(NPE), DF(NDIM, NPE)
DATA XNODE/-1.0D0, 2*I.0D0, 2*-I.0D0, 2*I.0D0, -I.0D0, 2*-I.0D0, 2*I.0D0,
1, 2*-I.0D0, 2*I.0D0, 4*-I.0D0, 4*I.0D0,
1, 2*1.0D0, 2*1.0D0, 2*1.0D0, -1.0D0, 2*-1.0D0, 2*1.0D0, -1.0D0,
2*1.0D0, 2*1.0D0, 4*-1.0D0, 4*1.0D0/
FCK(A, B, C) = 0.125*A*B*C
DO 20 I=1, NPE
XNP1 = XYZ(1) * XNODE(I, 1) + 1.0
YNP1 = XYZ(2) * XNODE(I, 2) + 1.0
ZNP1 = XYZ(3) * XNODE(I, 3) + 1.0
SUBROUTINE SURFGM(K1,KG1,ELXYZ,DSURF,GNORM,NBS,NGP,NPE,NDIM)

C
C    GNORM(I,J,K,L) ... I-TH COMPONENT OF 'NORMAL*DS' ON J-TH BOUNDARY
C    SURFACE AT (K,L) GAUSS POINT

C IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION ELXYZ (NPE,NDIM) , DSURF (NDIM,NPE,6,NGP,NGP) , GC (3,3),
*    GNORM (NDIM,NBS,NGP,NGP)

K0 = (K1+1)/2
K2=K0+1
IF (K2.EQ.4) K2=1
K3=K2+1
IF (K2.EQ.3) K3=1
DO 200 NGPI=1,NGP
    DO 200 NGPK=1,NGP
        CALL GCSURF (GC, DSURF, ELXYZ, NGPI, NGPK, NGP, K1, NDIM, NPE)
        DO 100 I=1,NDIM
            Ii=I+1
            IF (I1.EQ.4) I1=1
            I2=Ii+1
            IF (I2.EQ.3) I2=I
            100 GNORM(I, KG1,NGPI,NGPK) = (GC (Ii, K2) *GC (I2,K3) -GC (Ii, K3) *GC (I2, K2) )
        1     * (-1)**Ki
    200 CONTINUE
RETURN
END

***************************************************************
* SUBROUTINE TADMSH(X,Y,Z,IBNDC,KELSUR,NOD,NSURF,NNM,NBS,NDF,NPE, NPE)
***************************************************************
*
* MESH GENERATOR FOR TURN AROUND DUCT.
* *
* PURPOSE : TO GENERATE A THREE DIMENSIONAL MESH FOR A TURN AROUND
* DUCT. THE ELEMENT LIBRARY HAS THREE TYPES OF ELEMENTS
* VIZ. 8-NODED, 20 NODED, AND 27 NODED BRICK ELEMENTS.
*
*
*             /       /
*             |       |
*             |       | FACE 5 (BACK)
*             |       |
*             |       |           ETA
*             |       |
*             |       | FACE 3 -----
*             |       | (L. SIDE)
*             |       |           ZETA XI
*             |       | FACE 1 /
*             |       | (TOP)  /
*             |       | FACE 4 ---- FACE 4 / |
*             |       | (R. SIDE) ZETA XI
*             |       | FACE 2 (BOTTOM)
*             |       |
*             |       | FACE 6 |
LIST OF VARIABLES:

- NX1 = NUMBER OF DIVISIONS IN FLOW DIRN. IN PART 1 (INLET)
- NX2 = NUMBER OF DIVISIONS IN FLOW DIRN. IN PART 2 (CURVE)
- NX3 = NUMBER OF DIVISIONS IN FLOW DIRN. IN PART 3 (OUTLET)
- NY = NUMBER OF DIVISIONS IN RADIAL DIRECTION:
- NZ = NUMBER OF DIVISIONS IN Z-DIRECTION:
- NPE = NODES PER ELEMENT (8 OR 20 OR 27)
- NOD (NNM, NPE) = CONNECTIVITY MATRIX
- IEL = ELEMENT TYPE (1 = LINEAR (8 NODED); 2 = QUADRATIC)
- R1 = INNER RADIUS OF THE CURVE.
- R2 = OUTER RADIUS OF THE CURVE.
- X0 = X-COORDINATE OF FIRST NODE IN X-Y-Z PLANE.
- Y0 = Y-COORDINATE OF FIRST NODE IN X-Y-Z PLANE.
- Z0 = Z-COORDINATE OF FIRST NODE IN X-Y-Z PLANE
- X (NNM) = ARRAY CONTAINING X-COORDINATES OF NODES.
- Y (NNM) = ARRAY CONTAINING Y-COORDINATES OF NODES.
- Z (NNM) = ARRAY CONTAINING Z-COORDINATES OF NODES.
NDS = NYY*((NX1 + NX2 + NX3 + 1)*(NZ+1)) +
    (NY+1)*((NX1+NX2+NX3+1) + (NZ+1)*(NX1+NX2+NX3))
ELSE
NDS = NYY*(NXX1 + NXX2 + NXX3)*NZZ
END IF

IF (NDS .NE. NNM .OR. NEM.NE.NELM) THEN
    WRITE (6,999) NNM, NDS, NEM, NELM
    STOP
ENDIF

NTX = IEL*NX1 + 1
NTXX = IEL*NX2
NTXXX = IEL*NX3
NTXT = NTX + NTXX
NTXTT = NTXT + NTXXX

COMPUTE THE NODAL COORDINATES IN SECTION 1 (STRAIGHT INLET)

NTY = IEL*NY + 1
NTZ = IEL*NZ + 1
NY1 = (IEL-I)*NY + 1
IIX = 0
L = 0
DO 1050 IX = i, NTX
    IF(NPE .EQ. 20) THEN
        MODY = MOD (IX,2)
    ELSE
        MODY = 1
    END IF
    ZC = Z0
    ANGLE = PHI
    IF(MODY .EQ. i) THEN
        IF(NPE .EQ. 20) THEN
            I = (NYY*(NZ+I) + (NY+I)*(NZZ))*IIX
        ELSE
            I = NYY*(IX - I)*NZZ
        END IF
        DO 1020 IZ = i, NTZ
            IF(NPE .EQ. 20) THEN
                MODZ = MOD(IZ,2)
            ELSE
                MODZ = 1
            END IF
            IF(MODZ .EQ. i) THEN
                I = I + 1
                X(1) = X0
                Y(I) = RZ*COS(ANGLE)
                Z(I) = RZ*SIN(ANGLE)
                DO i000 IY = i, NTY-I
                    I = I + I
                    X(1) = X0
                    Y(I) = (Y(I-I) + DY(IY))*COS(ANGLE)
                    Z(I) = (Y(I-I) + DY(IY))*SIN(ANGLE)
                CONTINUE
            ELSE
                I = I + 1
                X(I) = X0
                Y(I) = Y0*COS(ANGLE)
        ELSE
            I = I + 1
            X(I) = X0
            Y(I) = Y0*COS(ANGLE)
    ELSE
        I = I + 1
        X(I) = X0
        Y(I) = Y0*COS(ANGLE)
    END IF

DO 1020 IZ = 1, NTZ
    IF(NPE .EQ. 20) THEN
        MODZ = MOD(IZ,2)
    ELSE
        MODZ = 1
    END IF
    IF(MODZ .EQ. 1) THEN
        I = I + 1
        X(I) = X0
        Y(I) = RZ*COS(ANGLE)
        Z(I) = RZ*SIN(ANGLE)
        DO 1000 IY = 1, NTY-1
            I = I + 1
            X(I) = X0
            Y(I) = (Y(I-1) + DY(IY))*COS(ANGLE)
            Z(I) = (Y(I-1) + DY(IY))*SIN(ANGLE)
        CONTINUE
    ELSE
        I = I + 1
        X(I) = X0
        Y(I) = Y0*COS(ANGLE)
    END IF
* 
* DO 1010 IY = 1, (NTY-NY1) 
  I = I + 1 
  K = 2*IY - 1 
  X(I) = X0 
  Y(I) = (RZ + DY(K) + DY(K+1))*COS(ANGLE) 
  Z(I) = (RZ + DY(K) + DY(K+1))*SIN(ANGLE) 
1010 CONTINUE
* 
* IF(IZ .LT. NTZ) ANGLE = ANGLE + ANGINC
* IIX = IIX + 1
* ELSE
* DO 1040 IZ = 1, (NZ+1) 
  I = I + 1 
  M = 2*IZ - 1 
  X(I) = X0 
  Y(I) = RZ*COS(ANGLE) 
  Z(I) = RZ*SIN(ANGLE) 
* 
* DO 1030 IY = 1, (NTY-NY1) 
  I = I + 1 
  K = 2*IX - 1 
  X(I) = X0 
  Y(I) = (RZ + DY(K) + DY(K+I))*COS(ANGLE) 
  Z(I) = (RZ + DY(K) + DY(K+I))*SIN(ANGLE) 
1030 CONTINUE
* ANGLE = ANGLE + ANGINC
1040 CONTINUE
* 
* END IF
* IF(IX .LT. NTX) X0 = X0 - DX1(IX)
* 1050 CONTINUE
* 
* COMPUTE THE NODAL COORDINATES IN THE CURVED SECTION:
* 
* NXPT1 = NTX + 1 
* THINC = PI/NXX2 
* THETA = PI + THINC 
* YC = Y0 + RZ 
* 
* DO 1110 IX = NXPT1, NTXT 
* IF(NPE .EQ. 20) THEN 
* MODY = MOD(IX,2) 
* ELSE 
* MODY = 1 
* END IF 
* 
* ZC = 20 
* ANGLE = PHI 
* IF(MODY .EQ. 1) THEN 
* DO 1080 IZ = 1, NTZ 
* IF(NPE .EQ. 20) THEN 
* MODZ = MOD(IZ,2) 
* ELSE 
* MODZ = 1 
* END IF 
* IF(MODZ .EQ. 1) THEN 
* I = I + 1 
* ELSE 
*
\[
X(I) = X_0 + R_2 \sin(\Theta) \\
Y(I) = (Y_C + R_2 \cos(\Theta)) \cos(\gamma) \\
Z(I) = (Y_C + R_2 \cos(\Theta)) \sin(\gamma)
\]

\[
DYY = 0.0D0 \\
DO 1060 IY = 1, NTY-1 \\
\quad I = I + 1 \\
\quad DYY = DYY + DY(IY) \\
\quad X(I) = X_0 + (R_2 - DYY) \sin(\Theta) \\
\quad Y(I) = (Y_C + (R_2-DYY) \cos(\Theta)) \cos(\gamma) \\
\quad Z(I) = (Y_C + (R_2-DYY) \cos(\Theta)) \sin(\gamma)
\]

\[
1060 \quad \text{CONTINUE}
\]

ELSE

\[
I = I + 1 \\
X(I) = X_0 + R_2 \sin(\Theta) \\
Y(I) = (Y_C + R_2 \cos(\Theta)) \cos(\gamma) \\
Z(I) = (Y_C + R_2 \cos(\Theta)) \sin(\gamma)
\]

\[
DYY = 0.0D0 \\
DO 1070 IY = 1, (NTY-NYI) \\
\quad I = I + 1 \\
\quad K = 2*IY - 1 \\
\quad DYY = DYY + DY(K) + DY(K+1) \\
\quad X(I) = X_0 + (R_2 - DYY) \sin(\Theta) \\
\quad Y(I) = (Y_C + (R_2-DYY) \cos(\Theta)) \cos(\gamma) \\
\quad Z(I) = (Y_C + (R_2-DYY) \cos(\Theta)) \sin(\gamma)
\]

\[
1070 \quad \text{CONTINUE}
\]

END IF

IF(IY .LT. NTZ) ANGLE = ANGLE + ANGINC

\[
1080 \quad \text{CONTINUE}
\]

ELSE

\[
IIX = IIX + 1
\]

* 

DO 1100 IZ = 1, (NZ+1) \\
\quad I = I + 1 \\
\quad M = 2*IZ - 1 \\
\quad X(I) = X_0 + R_2 \sin(\Theta) \\
\quad Y(I) = (Y_C + R_2 \cos(\Theta)) \cos(\gamma) \\
\quad Z(I) = (Y_C + R_2 \cos(\Theta)) \sin(\gamma)
\]

\[
DYY = 0.0D0 \\
DO 1090 IY = 1, (NTY-NYI) \\
\quad I = I + 1 \\
\quad K = 2*IY - 1 \\
\quad DYY = DYY + DY(K) + DY(K+1) \\
\quad X(I) = X_0 + (R_2 - DYY) \sin(\Theta) \\
\quad Y(I) = (Y_C + (R_2-DYY) \cos(\Theta)) \cos(\gamma) \\
\quad Z(I) = (Y_C + (R_2-DYY) \cos(\Theta)) \sin(\gamma)
\]

\[
1090 \quad \text{CONTINUE}
\]

\[
1100 \quad \text{CONTINUE}
\]

END IF

\[
\text{THETA} = \text{THETA} + \text{THINC}
\]

\[
1110 \quad \text{CONTINUE}
\]

* 

COMPUTE THE NODAL COORDINATES IN SECTION 3 (STRAIGHT OUTLET)

\[
\text{NTXP11} = \text{NTXT} + 1 \\
\text{YO} = \text{Y0} + 2.0*R_2 \cos(\Phi) \\
\text{J} = 0
\]

* 

DO 1170 IX = NTXP11, NTXTT
IF(NPE .EQ. 20) THEN
  MODY = MOD(IX,2)
ELSE
  MODY = 1
END IF

J = J + 1
X0 = X0 + DX3(J)
ZC = Z0 + 2.0*R2*SIN(PHI)
ANGLE = PHI

IF(MODY .EQ. 1) THEN
  DO 1140 IZ = 1, NTZ
      IF(NPE .EQ. 20) THEN
          MODZ = MOD(IZ,2)
      ELSE
          MODZ = 1
      END IF
      IF(MODY .EQ. I) THEN
        DO 1140 IY = I, NTY-I
            IF(NPE .EQ. 20) THEN
                MODY = MOD(IY,2)
            ELSE
                MODY = 1
            END IF
            IF(MODY .EQ. I) THEN
              I = I + 1
              X(I) = X0
              Y(I) = Y0*COS(ANGLE)
              Z(I) = Y0*SIN(ANGLE)
            ELSE
              I = I + 1
              X(I) = X0
              Y(I) = Y0*COS(ANGLE)
              Z(I) = Y0*SIN(ANGLE)
            END IF
        1140
      ELSE
        I = I + 1
        X(I) = X0
        Y(I) = Y0*COS(ANGLE)
        Z(I) = Y0*SIN(ANGLE)
        DO 1130 IY = I, (NTY-NY1)
            I = I + 1
            K = 2*IY - 1
            X(I) = X0
            Y(I) = (RZ+2*R2-DY(K)-DY(K+1))*COS(ANGLE)
            Z(I) = (RZ+2*R2-DY(K)-DY(K+1))*SIN(ANGLE)
        1130
      END IF
  ELSE
    I = I + 1
    X(I) = X0
    Y(I) = Y0*COS(ANGLE)
    Z(I) = X0*SIN(ANGLE)
    DO 1150 IY = I, (NTY-NY1)
        I = I + 1
        K = 2*IY - 1
        X(I) = X0
        Y(I) = (RZ+2*R2-DY(K)-DY(K+1))*COS(ANGLE)
        Z(I) = (RZ+2*R2-DY(K)-DY(K+1))*SIN(ANGLE)
    1150
  END IF
END IF

IF(IZ .LT. NTZ) ANGLE = ANGLE + ANGINC
1140 CONTINUE

IIX = IIX + 1

ELSE
  DO 1160 IZ = 1, (NZ+1)
    I = I + 1
    M = 2*IZ - 1
    X(I) = X0*COS(ANGLE)
    Y(I) = Y0
    Z(I) = X0*SIN(ANGLE)
  1160
  DO 1150 IY = 1, (NTY-NY1)
    I = I + 1
1150
\[ K = 2*IY - 1 \]
\[ X(I) = X0 \]
\[ Y(I) = (RZ+2*R2-DY(K)-DY(K+1)) \times \cos(\text{ANGLE}) \]
\[ Z(I) = (RZ+2*R2-DY(K)-DY(K+1)) \times \sin(\text{ANGLE}) \]

CONTINUE

\[ \text{ANGLE} = \text{ANGLE} + 2.*\text{ANGINC} \]

* END IF *

CONTINUE

DO 1175 I=I,NNM
\[ X(I)=0.0254*X(I) \]
\[ Y(I)=0.0254*Y(I) \]
\[ Z(I)=-0.0254*Z(I) \]

DETERMINE THE CONNECTIVITY MATRIX:

\[ \text{NX} = \text{NX1} + \text{NX2} + \text{NX3} \]

IF(NPE .EQ. 20) NTY = 3*NY + 2

DO 1200 IX = 1, NX
  DO 1190 IZ = 1, NZ
    DO 1180 IY = 1, NY
      I = IY + (IX-1)*NY*NZ + (IZ-1)*NY
      IF(NPE .EQ. 20) THEN
        NOD(I,1) = IEL*IY - (IEL-I) + (NYY*(NZ+1) + (NY+I)*NZZ)* (IX-1) + (IZ-1)* (NYY+NY)
      ELSE
        NOD(I,1) = IEL*IY - (IEL-I)*NYY*NZZ*IEL + (IZ-I)*IEL*NYY
      END IF
      NOD(I,2) = NYY*NZZ*IEL + NOD(I,1)
      NOD(I,3) = NOD(I,2) + IEL
      NOD(I,4) = NOD(I,1) + IEL
      IF(NPE .EQ. 20) THEN
        NOD(I,5) = NTY + NOD(I,1)
        NOD(I,6) = NYY*NZZ*IEL + NOD(I,5)
      ELSE
        NOD(I,5) = NYY + NOD(I,1)
      END IF
      NOD(I,7) = NOD(I,6) + IEL
      NOD(I,8) = NOD(I,5) + IEL
      IF(NPE .EQ. 20) THEN
        NOD(I,9) = NOD(I,1) + NYY*(NZ+1) + (NY+1)*NZZ + NOD(I,5)
        NOD(I,10) = NOD(I,2) + 1
        NOD(I,11) = NOD(I,9) + 1
        NOD(I,12) = NOD(I,1) + 1
        NOD(I,13) = NYY + NOD(I,1)
        NOD(I,14) = NYY + NOD(I,2)
        NOD(I,15) = NOD(I,14) + 1
        NOD(I,16) = NOD(I,13) + 1
        NOD(I,17) = NOD(I,5) + (NY + NY + 1)*NZ + (1-IY)
        NOD(I,18) = NOD(I,6) + 1
        NOD(I,19) = NOD(I,17) + 1
        NOD(I,20) = NOD(I,5) + 1
      ELSE
        NOD(I,9) = NOD(I,1) + NYY*(NZ+1) + (NY+1)*NZZ + (1-IY)
      END IF
ELSE IF (NPE .EQ. 27) THEN

NOD(I,9) = NOD(I,5) + NYY
NOD(I,10) = NOD(I,9) + NYY*NZZ*IEL
NOD(I,11) = NOD(I,10) + IEL
NOD(I,12) = NOD(I,9) + IEL
NOD(I,13) = NOD(I,1) + NYY*NZZ
NOD(I,14) = NOD(I,2) + 1
NOD(I,15) = NOD(I,13) + 2
NOD(I,16) = NOD(I,1) + 1
NOD(I,17) = NOD(I,5) + NYY*NZZ
NOD(I,18) = NOD(I,6) + 1
NOD(I,19) = NOD(I,17) + 2
NOD(I,20) = NOD(I,5) + 1
NOD(I,21) = NOD(I,9) + NYY*NZZ
NOD(I,22) = NOD(I,10) + 1
NOD(I,23) = NOD(I,21) + 2
NOD(I,24) = NOD(I,9) + 1
NOD(I,25) = NOD(I,13) + 1
NOD(I,26) = NOD(I,17) + 1
NOD(I,27) = NOD(I,21) + 1

END IF

COMPUTE THE NUMBER OF BOUNDARY SURFACES AND DETERMINE SURFACE INDICES

NSURF = 2*NX*(NY+NZ) + 2*NY*NZ

ELEMENT FLUX SURFACES AT THE INLET OF THE DUCT:

I = 0
NYZ = NY*NZ
DO 1210 IYZ = 1, NYZ
   I = I + 1
   KELSUR(I,1) = IYZ
   KELSUR(I,2) = 1
1210 CONTINUE

ELEMENT FLUX SURFACES AT THE SOLID SURFACE OF THE DUCT (OUTER):

DO 1220 IX = 1, NX
   DO 1220 IZ = 1, NZ
      I = I + 1
      ILL = (IX-1)*NY*NZ + (IZ-1)*NY + 1
      KELSUR(I,1) = ILL
      KELSUR(I,2) = 3
1220 CONTINUE

ELEMENT FLUX SURFACES AT THE SOLID SURFACE OF THE DUCT (INNER):

DO 1230 IX = 1, NX
   DO 1230 IZ = 1, NZ
      I = I + 1
      ILL = (IX-1)*NY*NZ + IZ*NY
      KELSUR(I,1) = ILL
      KELSUR(I,2) = 4
1230 CONTINUE

ELEMENT FLUX SURFACES AT SYMMETRY SURFACE OF THE DUCT (IZ = i):

DO 1240 IX = 1, NX
   DO 1240 IY = 1, NY
      I = I + 1
ILL = IY + (IX-1)*NY*NZ
KELSUR(I,1) = ILL
KELSUR(I,2) = 5
1240 CONTINUE
* *
ELEMENT FLUX SURFACES AT SYMMETRY SURFACE OF THE DUCT (IZ = NZ):
* *
DO 1250 IX = i, NX
   DO 1250 IY = i, NY
      I = I + 1
      ILL = IY + (IX-1)*NY*NZ + NY*(NZ-1)
      KELSUR(I,1) = ILL
      KELSUR(I,2) = 6
1250 CONTINUE
* *
ELEMENT FLUX SURFACES AT THE INLET OF THE DUCT:
* *
J = 0
DO 1260 IZ = i, NZ
   DO 1260 IY = I, NY
      J = J + 1
      I = I + 1
      ILL = (NX-I)*NY*NZ + J
      KELSUR(I,1) = ILL
      KELSUR(I,2) = 2
1260 CONTINUE
* *
* DETERMINE THE BOUNDARY CONDITIONS:
* *
NBNDC = 0
ND = 0
NXX = NX + 1
NYY = NY + 1
NZZ = NZ + 1
* *
DO 1212 I = I, NDS
   DO 1212 J = I, 5
      IBNDC(I,J) = 1
1212 CONTINUE
* *
* SPECIFY THE INLET BOUNDARY DEGREES OF FREEDOM
* *
DO 1280 ID = I, NYY
   DO 1270 JD = i, NZZ
      ND = ND + 1
      NBNDC = NBNDC + 1
      IBNDC(ND,2) = 0
      IBNDC(ND,3) = 0
      IBNDC(ND,4) = 0
      IBNDC(ND,5) = 0
1270 CONTINUE
1280 CONTINUE
* *
* SPECIFY THE SOLID-WALL BOUNDARY DEGREES OF FREEDOM
* *
DO 1300 KD = i, NX
   ND1 = (NYY*NZZ)*KD + 1
   DO 1290 JZ = I, NZZ
      ND = ND1 + (JZ-1)*NYY
      NBNDC = NBNDC + 1
      IBNDC(ND,2) = 0
      IBNDC(ND,3) = 0
      IBNDC(ND,4) = 0
      IBNDC(ND,5) = 0
1290 CONTINUE
1300 CONTINUE
*
CC
1290 CONTINUE
CC
1300 CONTINUE
* SPECIFY THE EXIT BOUNDARY DEGREES OF FREEDOM *
CC
NBD1 = NYY*NZZ*NX
DO 1320 I = 1, NZZ
    NBD = NBD1 + (I-1)*NYY
    DO 1310 J = 1, NYY
        NBD = NBD + 1
        NBNDC = NBNDC + 1
        IBNDC(NBD,5) = 0
        IBNDC(NBD,3) = 0
        IBNDC(NBD,4) = 0
    CONTINUE
CONTINUE
1310 CONTINUE
1320 CONTINUE
C RETURN
999 FORMAT(/,5X,'***** THE PARAMETERS NNM AND NEM SENT FROM THE MAIN
# DO NOT COINCIDE WITH THOSE GENERATED IN TADMOSH *****',/5X,'*****
# THE PROGRAM IS TERMINATED *****',/5X,'NNM,NDS,NEM,NEIM =','4I5)
END
FLOW IN A TURN-AROUND-DUCT (15X8X2 MESH)
1 1 02 100 05. 1.0 0.8
1.79E-03 293.0 110.0 287.0 1.402 0.72 0.1 1.205
3 8 4 8 2 1 8 1.0 3.0 33.0 9.0 0.0 2.0
20.0 8.0 2.0
0.5 1.5 8.0 20.0
0.1 0.2 0.3 0.4 0.4 0.3 0.2 0.1