Finite Element Solver for
3-D Compressible Viscous Flows

K. C. Reddy and J. N. Reddy

The University of Tennessee Space Institute
Tullahoma, Tennessee 37388

Interim Report of Contract No. NAS8-36555
For Period: April 1986 – October 1986

Submitted to

NASA/MSFC
Marshall Space Flight Center, AL 35812

by

The University of Tennessee Space Institute
Tullahoma, Tennessee 37388

December 1986
# Table of Contents

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

2. Technical Approach ........................................ 4
   2.1 Governing Equations .................................... 4
   2.2 Finite Element Model ................................... 6
   2.3 Locally Implicit Approximations ......................... 11

3. References .................................................. 12

Appendix I Finite Element Equations for Navier–Stokes Equations ....... 13

Appendix II 3-D Finite Element Navier–Stokes Code .................... 18
1. INTRODUCTION

The space shuttle main engine (SSME) has extremely complex internal flow structure. The geometry of the flow domain is three-dimensional with complicated topology. The flow is compressible, viscous and turbulent with large gradients in flow quantities and regions of recirculations. In recent years computer codes are being developed\(^{(1-4)}\) to solve the flow equations in different regions of the SSME such as the hot gas manifold (HGM) region. The analysis of the flow field in SSME involves several tedious steps. One is the geometrical modelling of the particular zone of the SSME being studied. It is usually available in the form of engineering drawings, in terms of algebraic equations for different pieces of the surfaces or in a CAD (computer aided design) system. Accessing the geometry definition, digitizing it and developing surface interpolations suitable for an interior grid generator requires considerable amount of manual effort. There are several types of grid generators available with some general-purpose finite element programs, such as NASTRAN, ADINA, ABQUS, etc. However, these programs require considerable amount of effort on the part of the user to input the geometry to the grid generators; also, the grid generated by those programs are not always the most appropriate grids for the flows being modelled. Next, an efficient and robust computational scheme for solving 3D Navier-Stokes equations has to be implemented for this class of problems. Post processing software has to be adapted to visualize and analyze the computed 3D flow field. Different elements of the above process have been studied in the past and other parts are yet to be developed. The current report discusses the progress made in a project to develop software for the analysis of the flow in the space shuttle main engine and similar complex internal flows.

A CFD code for practical applications should have the following features. It should be reasonably accurate for the class of problems it is designed to solve, with grids that can be accommodated on the present day computers. It should be robust in the sense that it is numerically stable for a broad range of initial and boundary conditions and geometrical parameters, and tolerate some variations in the grid resolution and structure. It should be computationally efficient for obtaining accurate solutions with reasonable computational and human resources. Standard of efficiency, however, is relative and it can only be measured against the current CFD software or which can be foreseen in the immediate future. Another important aspect of a CFD code is its usability, as to how much effort a user has to expend to solve practical problems with it.
For computing the viscous compressible flow inside the main engine where the flow undergoes complex turns through various chambers and ducts, it is necessary to discretize the physical space with several competing requirements. The geometry of the internal surfaces is typically represented in a CAD system or in some equivalent form by the designer. The surface data representation should be interfaced with suitable interpolation software. The refined spline surface representation of the flow boundaries will be the input for the grid generation routines. The topology of the grid structure depends on the flow solver algorithm to be used. Finite difference codes usually impose constraints on the grid structure such as the separability of the indices for efficient computational procedures, while the finite element method can be implemented with less stringent requirements on the grid structure. The grid should provide reasonable resolution of the flow field within the limits of the grid selected by the user. This requires providing more grid points and/or special methods in regions of large gradients of flow quantities, such as the viscous zones near solid boundaries. The grid should meet certain smoothness requirements so that the metrics of the curvilinear grid can be computed numerically and the computed metrics are nonsingular. Unreasonably skewed grid cells or elements, and singular points in the grid where the local transformation of the physical space to computational space has very small or very large Jacobians, should be avoided if at all possible. Otherwise such grids will require special handling by the flow solver algorithm and also may give rise to numerical inaccuracies and instabilities.

There are several grid generation techniques and special purpose codes which can generate reasonable grids for simple two-dimensional and three-dimensional geometries, for both internal and external flows. These techniques fall under two classes: algebraic generators and elliptic generators. Algebraic generators use various interpolation and stretching functions while elliptic generators solve a set of elliptic partial differential equations. While both techniques are effective for simple geometric regions, it is usually difficult to use them to develop a composite grid over a complex internal flow domain. Finite element community have developed extensive amount of software for generating algebraic grid suitable for finite element solvers. For example, NASTRAN (a general purpose finite element program primarily developed for structural analysis) contains grid generators for 2D and 3D structures. Also, the program PATRAN (developed by PDA Engineering) contains 2D and 3D grid generators and pre- and post processing capabilities. In the current project some parts of the software such as PATRAN will be adapted and developed to generate body conforming, curvilinear finite element meshes of the flow domains inside the SSME.
Computation of the flow field inside the space shuttle main engine requires the application of the state-of-the-art CFD technology. Several computer codes\(^{(1-4)}\) are under development to solve three dimensional Navier-Stokes equations with different turbulence models for analyzing the SSME internal flow, such as the flow through the how gas manifold (HGM). The computational methods\(^{(5-6)}\) 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 use 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\(^{(6)}\). 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. Desired degree of approximation (e.g., linear, quadratic, etc.) can be easily and routinely specified without rewriting the whole or parts of the program. The third feature is that the relationship (i.e., the algebraic equations) between the solution and its dual variables (i.e., velocities and forces) 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 implicit finite element scheme with suitable dissipation terms for stability is being developed. A relaxation procedure, known as the locally implicit scheme is being developed to solve the coupled set of algebraic equations efficiently.

In the following sections we discuss the technical approach to the development of the finite element scheme and the relaxation procedure. Appendix I contains the details of the equations derived and Appendix II has a listing of the three dimensional finite element code for the compressible Navier-Stokes equations. Future reports will discuss the numerical results for specific problems.

2. TECHNICAL APPROACH

2.1 GOVERNING EQUATIONS

In an Eulerian description, used most extensively in fluid dynamics, the coordinate system is fixed in space rather than in the body, and measurements of density, velocity, pressure, etc. are made for the material particle that happens to be in a given location at that particular time. The basic equations of a continuous medium in the Eulerian description are:

**Continuity Equation.** - The law of conservation of mass leads to

\[
\frac{\partial}{\partial t} (\rho) + \nabla \cdot (\rho \mathbf{u}) = 0
\]

where \( \rho \) is the density of the medium, \( \mathbf{u} \) is the velocity vector and \( \mathbf{x} = (x_1, x_2, x_3) \) the spatial coordinates.
Equations of Motion. – The law of balance of linear momentum leads to the celebrated Eulerian equation of motion,

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) = \nabla \cdot \mathbf{\sigma} + \mathbf{F}$$

(2)

Here $\mathbf{F}$ the body force vector (measured per unit volume) and $\mathbf{\sigma}$ the total stress tensor, which can be divided into hydrostatic and viscous parts:

$$\mathbf{\sigma} = -p \mathbf{I} + \mathbf{r}$$

(3)

Here $p$ denotes the hydrostatic pressure, $\mathbf{r}$ the viscous (or shear) stress tensor, and $\mathbf{I}$ denotes the unit tensor.

An application of the law of balance of angular momentum and neglect of microstructural effects such as couple stresses lead to the symmetry of stress tensor,

$$\sigma_{ij} = \sigma_{ji}, \quad \tau_{ij} = \tau_{ji} \quad (\mathbf{\sigma} = \mathbf{\sigma}^T)$$

(4)

Energy Equation. – The law of conservation of energy (the first law of thermodynamics) leads to

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot (\rho e \mathbf{v}) = \nabla \cdot (\mathbf{\sigma} \cdot \mathbf{v}) + \mathbf{F} \cdot \mathbf{v} + \rho S - \nabla \cdot \mathbf{q}$$

(5)

where $e$ is the total energy per unit mass,

$$e = \varepsilon + \frac{1}{2} \mathbf{v} \cdot \mathbf{v}$$

$\varepsilon$ being the specific internal energy, $S$ is the rate of internal heat generation per unit mass, and $\mathbf{q}$ is the heat flux vector or the rate of heat flow per unit area across the surface in the direction of its unit outward normal.

Constitutive Equations. – The thermodynamic pressure $p$ is related to the specific internal energy $\varepsilon$ and the density $\rho$ through an equation of state,

$$p = p(\varepsilon, \rho)$$

(6)

and the viscous stress is related to the deformation rate tensor $\mathbf{d}$ through a constitutive equation of the form

$$\mathbf{r} = \mathbf{r}(\mathbf{d}, \varepsilon)$$

(7)
where
\[ d = \frac{1}{2} \left[ \nabla v + (\nabla v)^T \right] \]  
and \( e \) is the tensor of viscosities.

For isotropic fluids obeying linear stress-strain relations (i.e. Newtonian fluids) we have
\[ \tau_{ij} = 2\mu d_{ij} \]  
where \( \mu \) is the viscosity.

**Initial Conditions.** At time \( t = 0 \), values of all the dependent variables \((\rho, v, e, p)\) must be specified in the entire domain. It is not essential to specify all of these quantities at the same set of points.

**Boundary Conditions.** Depending on the type of the boundary (e.g., rigid boundary, free surface, interface, plane of symmetry, etc.), there are different kinds of boundary conditions in a problem. At a rigid boundary, the normal component of the particle velocity must coincide with the normal component of the velocity of the rigid boundary. For a fixed (in time) boundary, the normal component of the particle velocity must be zero at that boundary. A plane of symmetry can be interpreted as a fixed boundary. On a free surface, the pressure must vanish. At an interface (and at a contact discontinuity) the pressure and the normal component of particle velocity must be continuous, and the density, internal energy and the tangential component of particle velocity may be discontinuous (i.e. jumps may occur). Across moving shock fronts, the Rankine–Hugoniot relations must be satisfied.

### 2.2 FINITE ELEMENT MODEL

Writing the governing equations in terms of the velocities, pressure, density and internal energy, we obtain
\[ \frac{\partial \rho}{\partial t} - \nabla \cdot (\rho v) = 0 \]
\[ \frac{\partial}{\partial t} (\rho v) + \nabla \cdot (\rho vv) - \nabla p = \mu \nabla \cdot d + F \]
\[ \frac{\partial}{\partial t} (\rho e) + \nabla \cdot (\rho ve) + \nabla \cdot (pv) = \mu \nabla \cdot (d \cdot v) + F \cdot v - \nabla \cdot q \]  
and \( p \) is given by the equation of state. If we assume that the body force, heat flux, and the internal heat generation are zero, the last two terms in the energy equation dropout.
For simplicity and computational convenience, we denote

\[ \rho v = V, \quad \rho e = E \]

so that (10) become

\[ \frac{\partial}{\partial t} (\rho) + \nabla \cdot V = 0 \]
\[ \frac{\partial}{\partial t} (V) + \nabla \cdot (\mu V) + vP = \mu \nabla \cdot (d \cdot v) \]

We seek approximate solutions to Eq. (11) using the finite element method.

**Spatial Approximation.** - Finite element approximations to Eq. (11) are sought over a typical element \( \Omega^e \):

\[
\begin{align*}
\rho &= \sum_{j=1}^{N} \rho_j \psi_j(x) \\
V_i &= \sum_{j=1}^{N} V_i^j(t) \psi_j(x) \\
E &= \sum_{j=1}^{N} E_j(t) \psi_j(x)
\end{align*}
\]

(12)

where \( \psi_j(x) \) are the interpolation functions in space, \( \rho_j, V_i^j, \) and \( E_j \) are the unknown, time-dependent, nodal values to be determined. In Eq. (12) we have assumed for simplicity the same type (linear or quadratic) of interpolation functions for all the variables. The Galerkin approximation amounts to seeking solutions to Eq. (11) in the form (12) by making the errors in Eq. (11) orthogonal to the trial functions. This leads to the following local set of nonlinear ordinary differential equations in time.

\[
\begin{align*}
[A] \{\dot{\rho}\} + [B] \{V\} &= 0 \\
[A] \{\dot{V}\} + [N] \{V\} &= \{Q\} \\
[A] \{\dot{E}\} + [M] \{E\} &= \{R\}
\end{align*}
\]

(13)

Here the superposed dot denote total differentiation with respect to time, and

\[
A_{ij} = \int_{\Omega^e} \psi_\alpha \psi_\beta \, d\bar{x}, \quad B_{ij} = \int_{\Omega^e} \psi_i \frac{\partial \psi_j}{\partial \bar{x}_k} \, d\bar{x}
\]
\[ N_{ij} = \int_{\Omega^e} \psi_i \sum_{k=1}^{3} \frac{\partial}{\partial x_k} (v_k \psi_j) \, d\mathbf{x} + \int_{\Omega^e} \mu \nabla \psi_i \cdot \nabla \psi_j \, d\mathbf{x}, \]

\[ M_{ij} = \int_{\Omega^e} \psi_i \sum_{k=1}^{3} \frac{\partial}{\partial x_k} (v_k \psi_j) \, d\mathbf{x}, \]

\[ Q_{ik} = -\int_{\Omega^e} \psi_i \frac{\partial p}{\partial x_k} \, d\mathbf{x}, \quad R_i = -\int_{\Omega^e} \psi_i \sum_{k=1}^{3} \frac{\partial}{\partial x_k} (p v_k) \, d\mathbf{x} + \int_{\Omega^e} \mu \nabla \psi_i \, d\mathbf{x} \quad (14) \]

where \( \int_{\Omega^e} \) denotes integration over the element volume.

Equations (13) are to be further approximated (or numerically integrated with respect to time) to obtain a set of simultaneous algebraic equations.

**Temporal Approximations.** – Equations (13) are of the general form

\[ [A] \{ \dot{U} \} + [B] \{ U \} = \{ Q \}, \quad (15) \]

We approximate \( U(t) \) by

\[ U(t) = \sum_{j=1}^{n} U_j \phi_j(t), \quad m = 1, 2, \ldots, M \quad (16) \]

where \( \phi_j(t) \) are approximation functions in time. Here we assume that \( \phi_j \) are linear in \( t \) (i.e., \( n = 2 \)):

\[ \phi_1(t) = \left( 1 - \frac{t}{\Delta t} \right), \quad \phi_2(t) = \frac{t}{\Delta t}, \quad 0 \leq t \leq \Delta t \]

where \( \Delta t \) denotes the time increment. Then the time derivative of \( U \) is given by

\[ \dot{U} = (U_2 - U_1) / \Delta t \quad (17) \]

It can be readily interpreted that \( U_1 \) is the value of \( U \) at time \( t = n(\Delta t) \), and \( U_2 \) is the value of \( U \) at \( t = (n + 1)\Delta t \). Substituting Eq. (16) and (17) into Eq. (15), multiplying with \( \phi_2(t) \) and integrating over 0 to \( \Delta t \), we obtain

\[ \begin{bmatrix} A + \frac{2}{3} \Delta t B \end{bmatrix} \{ U_{n+1} \} = \Delta t \{ Q \} - \left[ A - \frac{\Delta t}{3} B \right] \{ U_n \} \quad (18) \]

Thus the unknown vector \( \{ U_{n+1} \} \) can be solved in terms of the known vector \( \{ U_n \} \). It should be noted that the temporal approximations (18) can be applied to the local set (13). There are other methods of time integration which can be incorporated into the code.
Equations (18) can be assembled in the usual manner to obtain the global equations, which must be solved iteratively (after imposing the initial and boundary conditions of the problem) for the nodal values, as the resulting algebraic equations are nonlinear. A flow chart of the computer program based on the formulation presented above is shown in Fig. 1.
CALL MESH to generate the mesh, and initial and boundary conditions

TIME = 0.0

DO ITM = 1, NTM

TIME = TIME + DELT

DO LL = 1, IU

DO N = 1, NET

CALL MESH to generate the mesh, and initial and boundary conditions

CALL FRISUR to update the coordinates

CALL a SUBROUTINE to calculate pressure

CALL BDUNSM to solve the system

Impose boundary conditions on global matrix

Assemble element matrix

CALL TEMPRL for temporal approximation

CALL SPatial to generate element matrices

SUBROUTINE MESH

SUBROUTINE FRISUR

SUBROUTINE ALAMOS

SUBROUTINE TILTSN

SUBROUTINE BDUNSM

SUBROUTINE TEMPRL

SUBROUTINE SPatial

Fig. 1. Flow Chart of the Computer Program
2.3 LOCALLY IMPLICIT APPROXIMATIONS

For large problems, it is not possible to solve the global (linearized) equations by direct methods. An efficient iterative method of solution has been formulated and it is known as the locally implicit method\(^{(7)}\). This is based on a modified Gauss-Seidel iteration technique with a symmetric inner iteration.

The linearized equations (18) for an element \(j\) can be written in the form

\[
L_j \Delta U_j = Res \left( U^n \right) + \sum_{k \neq j} L_k \Delta U_k + Q \tag{19}
\]

where \(\Delta U_j = U_j^{n+1} - U_j^n\) and the summation on the right hand side of (19) is limited to the elements surrounding the element \(j\), with which the finite element equations over the element \(j\) are coupled with. Equations (19) are solved by an iteration

\[
LM_j \delta \Delta U_j = Res \left( U^n \right) + \sum_{k \neq j} L_k \Delta U_k^{(m)} - L_j \Delta U_j^{(m)} + Q \tag{20}
\]

where \(\Delta U_j^{m+1} = \Delta U_j^{(m)} + \delta \Delta U_j\), \(LM_j\) is a modification to the matrix \(L_j\) so as to achieve stability and rapid convergence of the iteration process. \(\Delta U_k^{(\cdot)}\) denotes either \(\Delta U_k^{(m+1)}\) or \(\Delta U_k^{(m)}\) depending on the latest available iterates for \(\Delta U_k\). The iteration process of the equation (20) is carried out starting at a different corner of the computational space for each iteration. Eight such iterations complete one symmetric modified Gauss-Seidel iteration per time step for 3-dimensional problems. This is a stable process with fast convergence properties in a local sense. It amounts to solving the equations (15) implicitly in a local sense for each node. It is not necessary to achieve full convergence at each time step if we need only the steady state solution. One symmetric sweep per time step is adequate. This process has been tested over a variety of model equations such as the 3-dimensional Poisson equation and one dimensional Burger's equation. The same procedure has also been shown to work for two dimensional Euler equations with finite volume discretizations and artificial dissipation terms.
3. REFERENCES


Appendix I

Finite Element equations for Navier–Stokes Equations

Variational formulation over an element for the Navier–Stokes equations in non-conservation form:

\[ 0 = \int_{\Omega^e} w_1 \left[ \frac{\partial \rho}{\partial t} + u \frac{\partial \rho}{\partial x} + v \frac{\partial \rho}{\partial y} + w \frac{\partial \rho}{\partial z} + \rho \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right] dV \]  \hspace{1cm} (1)

\[ 0 = \int_{\Omega^e} \left\{ \rho w_2 \frac{\partial u}{\partial t} + w_2 \rho \left( u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right) - p \frac{\partial w_2}{\partial x} + 2\mu \frac{\partial w_2}{\partial x} + 2\mu \frac{\partial w_2}{\partial x} \frac{\partial u}{\partial x} \right. \]
\[ \left. + \mu \frac{\partial w_2}{\partial y} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) + \mu \frac{\partial w_2}{\partial z} \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) - \lambda \frac{\partial w_2}{\partial x} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right\} dV \]
\[ - \int_{\Gamma^e} t_z w_2 ds \]  \hspace{1cm} (2)

\[ 0 = \int_{\Omega^e} \left\{ \rho w_3 \frac{\partial v}{\partial t} + w_3 \rho \left( u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} \right) - p \frac{\partial w_3}{\partial y} + \mu \frac{\partial w_3}{\partial y} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right. \]
\[ \left. + 2\mu \frac{\partial w_3}{\partial y} \frac{\partial v}{\partial x} + \mu \frac{\partial w_3}{\partial z} \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial x} \right) - \lambda \frac{\partial w_3}{\partial y} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right\} dV \]
\[ - \int_{\Gamma^e} t_y w_3 ds \]  \hspace{1cm} (3)

\[ 0 = \int_{\Omega^e} \left\{ \rho w_4 \frac{\partial w}{\partial t} + w_4 \rho \left( u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} \right) - p \frac{\partial w_4}{\partial z} + \mu \frac{\partial w_4}{\partial z} \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \right. \]
\[ \left. + \mu \frac{\partial w_4}{\partial y} \left( \frac{\partial w}{\partial z} + \frac{\partial w}{\partial y} \right) + 2\mu \frac{\partial w_4}{\partial z} \frac{\partial w}{\partial x} - \lambda \frac{\partial w_4}{\partial z} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right\} dV \]
\[ - \int_{\Gamma^e} t_z w_4 ds \]  \hspace{1cm} (4)

\[ 0 = \int_{\Omega^e} \left\{ \rho c_v w_5 \frac{\partial T}{\partial t} + \rho c_v w_5 \left( u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} + w \frac{\partial T}{\partial z} \right) - w_5 \rho Q \right. \]
\[ + k_x \frac{\partial w_5}{\partial x} \frac{\partial T}{\partial x} + k_y \frac{\partial w_5}{\partial y} \frac{\partial T}{\partial y} - k_z \frac{\partial w_5}{\partial z} \frac{\partial T}{\partial z} - w_5 \rho \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \]
\[ - w_5 \left( \frac{\partial u}{\partial x} \tau_{zz} + \frac{\partial u}{\partial y} \tau_{zy} + \frac{\partial u}{\partial z} \tau_{xz} + \frac{\partial v}{\partial x} \tau_{yy} + \frac{\partial v}{\partial y} \tau_{yz} + \frac{\partial w}{\partial x} \tau_{zx} + \frac{\partial w}{\partial y} \tau_{zy} + \frac{\partial w}{\partial z} \tau_{xz} \right) \}
\[ dV \]
\[ - \int_{\Gamma^e} q w_5 ds \]  \hspace{1cm} (5)
where
\[ t_x = \sigma_x n_x + \sigma_{xy} n_y + \sigma_{xz} n_z, \quad t_y = \sigma_{xy} n_x + \sigma_y n_y + \sigma_{yz} n_z \]
\[ t_z = \sigma_{xz} n_x + \sigma_{yz} n_y + \sigma_z n_z, \quad q = K_z \frac{\partial T}{\partial x} n_x + K_y \frac{\partial T}{\partial y} n_y + K_z \frac{\partial T}{\partial z} n_z \]

**FINITE ELEMENT FORMULATION**

Let \( \rho = \sum_{j=1}^{n} \rho_j \psi_j(x, y, z), \quad u = \sum_{j=1}^{n} U_j \psi_j(x, y, z), \) etc.

Equations (1) – (5) can be formulated as

\[
\begin{align*}
[M^1]{\dot{\rho}} + [K^1]{\rho} &= \{F^1\} \\
[M^2]{\dot{U}} + [K^2]{U} &= \{F^2\} \\
[M^2]{\dot{V}} + [K^3]{V} &= \{F^3\} \\
[M^2]{\dot{W}} + [K^4]{W} &= \{F^4\} \\
[M^3]{\dot{T}} + [K^5]{T} &= \{F^5\}
\end{align*}
\]

where

\[
M_{ij}^1 = \int_{\Omega} \psi_i \psi_j dV, \quad K_{ij}^1 = \int_{\Omega} \psi_i \left( u \frac{\partial \psi_j}{\partial x} + v \frac{\partial \psi_j}{\partial y} + w \frac{\partial \psi_j}{\partial z} \right) dV
\]

\[
F_i^1 = -\int_{\Omega} \rho \psi_i \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) dV
\]

\[
M_{ij}^2 = \int_{\Omega} \rho \psi_i \psi_j dV, \quad K_{ij}^2 = \int_{\Omega} \left[ \rho \psi_i \left( u \frac{\partial \psi_j}{\partial x} + v \frac{\partial \psi_j}{\partial y} + w \frac{\partial \psi_j}{\partial z} \right) + 2\mu \frac{\partial \psi_i \partial \psi_j}{\partial x} \frac{\partial \psi_i}{\partial x} \\
+ \mu \frac{\partial \psi_i \partial \psi_j}{\partial y} \frac{\partial \psi_i}{\partial y} + \mu \frac{\partial \psi_i \partial \psi_j}{\partial z} \frac{\partial \psi_i}{\partial z} - \lambda \frac{\partial \psi_i \partial \psi_j}{\partial x} \frac{\partial \psi_i}{\partial x} \right] dV
\]
\[ F_i^2 = \int_{\Omega^*} \left[ p \frac{\partial\psi_i}{\partial x} - \mu \left( \frac{\partial\psi_i}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial\psi_i}{\partial z} \frac{\partial w}{\partial x} \right) + \lambda \frac{\partial\psi_i}{\partial x} \left( \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right] dV \\
\qquad + \oint_{\Gamma^*} t_z \psi_i ds \]

\[ K_{ij}^3 = \int_{\Omega^*} \left[ \rho \psi_i \left( \frac{\partial\psi_j}{\partial x} + v \frac{\partial\psi_j}{\partial y} + w \frac{\partial\psi_j}{\partial z} \right) + \mu \left( \frac{\partial\psi_i}{\partial x} \frac{\partial \psi_j}{\partial y} + \frac{2 \partial\psi_i}{\partial y} \frac{\partial \psi_j}{\partial y} + \frac{\partial \psi_i}{\partial z} \frac{\partial \psi_j}{\partial z} \right) \right] \frac{\partial\psi_i}{\partial y} \bigg|_{y} dV \\
\qquad - \lambda \frac{\partial\psi_i}{\partial y} \bigg|_{y} dV \\
\]

\[ F_i^3 = \int_{\Omega^*} \left[ p \frac{\partial\psi_i}{\partial y} + \mu \left( \frac{\partial\psi_i}{\partial x} \frac{\partial u}{\partial y} + \frac{\partial\psi_i}{\partial z} \frac{\partial w}{\partial y} \right) - \lambda \frac{\partial\psi_i}{\partial y} \left( \frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} \right) \right] dV \\
\qquad - \oint_{\Gamma^*} t_y \psi_i ds \]

\[ K_{ij}^4 = \int_{\Omega^*} \left[ \psi_i \rho \left( u \frac{\partial\psi_i}{\partial x} + v \frac{\partial\psi_i}{\partial y} + w \frac{\partial\psi_i}{\partial z} \right) + \mu \left( \frac{\partial\psi_i}{\partial x} \frac{\partial \psi_j}{\partial y} + \frac{2 \partial\psi_i}{\partial y} \frac{\partial \psi_j}{\partial y} + \frac{\partial \psi_i}{\partial z} \frac{\partial \psi_j}{\partial z} \right) \right] \frac{\partial\psi_i}{\partial z} \bigg|_{z} dV \\
\qquad - \lambda \frac{\partial\psi_i}{\partial z} \bigg|_{z} dV \\
\]

\[ F_i^4 = \int_{\Omega^*} \left[ p \frac{\partial\psi_i}{\partial z} + \mu \left( \frac{\partial\psi_i}{\partial x} \frac{\partial u}{\partial z} + \frac{\partial\psi_i}{\partial y} \frac{\partial w}{\partial z} \right) - \lambda \frac{\partial\psi_i}{\partial z} \left( \frac{\partial u}{\partial x} + \frac{\partial w}{\partial y} \right) \right] dV \\
\qquad - \oint_{\Gamma^*} t_z \psi_i ds \]

\[ K_{ij}^5 = \int_{\Omega^*} \left[ \rho c_v \psi_i \left( u \frac{\partial\psi_j}{\partial x} + v \frac{\partial\psi_j}{\partial y} + w \frac{\partial\psi_j}{\partial z} \right) + K_z \frac{\partial\psi_i}{\partial x} \frac{\partial \psi_j}{\partial x} + K_y \frac{\partial\psi_i}{\partial y} \frac{\partial \psi_j}{\partial y} \right] dV \\
\qquad + K_z \frac{\partial\psi_i}{\partial z} \bigg|_{z} dV \\
\]

\[ F_i^5 = \int_{\Omega^*} \left[ \psi_i \rho Q - \psi_i \rho \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + \psi_i \left( \frac{\partial u}{\partial x} r_{zz} + \cdots \right) \right] dV \\
\qquad + \oint_{\Gamma^*} q \psi_i ds \]

\[ M_{ij}^3 = \int_{\Omega^*} \rho c_v \psi_i \psi_j dV \]
ALTERNATIVE (CONSERVATION) FORM OF EQUATIONS

Let \( \vec{V} = \rho \vec{v} \) (\( U = \rho u \), \( V = \rho v \), \( W = \rho w \))

\[ E = \rho \varepsilon, \quad \vec{f} = \vec{0}, \quad Q = 0 \]

The governing equations are

\[
\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) + \frac{\partial}{\partial z}(\rho w) = 0
\]

\[
\frac{\partial U}{\partial t} + \frac{\partial}{\partial x}(U u) + \frac{\partial}{\partial y}(U v) + \frac{\partial}{\partial z}(U w) = -\frac{\partial p}{\partial x} + \frac{\partial \sigma_x}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z}
\]

\[
\frac{\partial V}{\partial t} + \frac{\partial}{\partial x}(V u) + \frac{\partial}{\partial y}(V v) + \frac{\partial}{\partial z}(V w) = -\frac{\partial p}{\partial y} + \frac{\partial \sigma_y}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z}
\]

\[
\frac{\partial W}{\partial t} + \frac{\partial}{\partial x}(W u) + \frac{\partial}{\partial y}(W v) + \frac{\partial}{\partial z}(W w) = -\frac{\partial p}{\partial z} + \frac{\partial \sigma_z}{\partial x} + \frac{\partial \sigma_{xz}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z}
\]

\[
\frac{\partial E}{\partial t} + \frac{\partial}{\partial x}(u E) + \frac{\partial}{\partial y}(v E) + \frac{\partial}{\partial z}(w E) = -\vec{\nabla} \cdot \vec{q} + \vec{\sigma} : \vec{D}
\]

The finite-element equations are

\[
[M]\{ \dot{\rho} \} + [K]\{ \rho \} = \{ F^1 \}
\]

\[
[M]\{ \dot{U} \} + [K]\{ U \} = \{ F^2 \}
\]

\[
[M]\{ \dot{V} \} + [K]\{ V \} = \{ F^3 \}
\]

\[
[M]\{ \dot{W} \} + [K]\{ W \} = \{ F^4 \}
\]

\[
[M]\{ \dot{E} \} + [K]\{ E \} = \{ F^5 \}
\]

16
where

\[
M_{ij} = \int_{\Omega^*} \psi_i \psi_j \, dV, \quad K_{ij} = \int_{\Omega^*} \psi_i \left[ \frac{\partial}{\partial x} (u \psi_j) + \frac{\partial}{\partial y} (v \psi_j) + \frac{\partial}{\partial z} (w \psi_j) \right] \, dV
\]

\[
= \int_{\Omega^*} \psi_i \left[ u \frac{\partial \psi_j}{\partial x} + v \frac{\partial \psi_j}{\partial y} + w \frac{\partial \psi_j}{\partial z} + \psi_j \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right] \, dV
\]

\[
F_i^1 = 0, \quad F_i^2 = \int_{\Omega^*} \psi_i \left[ -\frac{\partial p}{\partial x} + \frac{\partial \sigma_{x x}}{\partial x} + \frac{\partial \sigma_{x y}}{\partial y} + \frac{\partial \sigma_{x z}}{\partial z} \right] \, dV
\]

\[
F_i^3 = \int_{\Omega^*} \psi_i \left[ -\frac{\partial p}{\partial y} + \frac{\partial \sigma_{y y}}{\partial x} + \frac{\partial \sigma_{y z}}{\partial y} + \frac{\partial \sigma_{y z}}{\partial z} \right] \, dV
\]

\[
F_i^4 = \int_{\Omega^*} \psi_i \left[ -\frac{\partial p}{\partial z} + \frac{\partial \sigma_{z z}}{\partial x} + \frac{\partial \sigma_{z x}}{\partial y} + \frac{\partial \sigma_{z z}}{\partial z} \right] \, dV
\]

\[
F_i^5 = \int_{\Omega^*} \left( -\vec{\nabla} \cdot \vec{q} + \vec{\sigma} : \vec{D} \right) \psi_i \, dV
\]

This formulation is a natural extension of the finite element model for inviscid flows and is applicable for compressible viscous flows from low subsonic to supersonic flows with suitable addition of stabilizing terms (artificial viscosity). For highly viscous, low Mach number internal flows there is no need for the addition of artificial viscosity. This formulation is coded in the computer program COMPR3D and is listed in Appendix II.
AN IN-CORE FINITE-ELEMENT ANALYSIS COMPUTER PROGRAM FOR THE ANALYSIS OF UNSTEADY NAVIER-STOKES EQUATIONS GOVERNING THE FLOW OF A VISCOS COMPRESSIBLE FLUID IN THREE-DIMENSIONAL ENCLOSURES. THE CONSERVATION FORM OF THE EQUATIONS IS USED TO DEVELOP THE FINITE ELEMENT MODEL. THE PROGRAM IS UNDER DEVELOPMENT BY J. N. REDDY, 505 CRANWELL CIRCLE, BLACKSBURG.

DESCRIPTION OF THE VARIABLES

AMU......VISCOSITY OF THE FLUID (=1.0/REYNALDS NUMBER HERE)
BETA......ACCELERATION PARAMETER IN THE NONLINEAR ITERATION
DT........TIME STEP FOR THE UNSTEADY ANALYSIS
DX,DY,DZ..ARRAYS OF COORDINATES OF THE NODES ALONG THE X, Y AND Z COORDINATES, RESPECTIVELY (FOR MESH ONLY)
CV........CONSTANT IN THE EQUATION OF STATE (GAS CONSTANT)
DRE.......ARRAY OF THE INCREMENTS OF REYNALDS NUMBER
EPS.......ERROR PERMITTED BETWEEN THE SOLUTIONS OF TWO CONSECUTIVE TIMES (A CHECK FOR STEADY STATE SOLUTION)
ELF.......ELEMENT SOURCE VECTOR
GSTIF......ASSEMBLED COEFFICIENT MATRIX; THE LAST COLUMN OF THE MATRIX CONTAINS THE SOURCE VECTOR BEFORE GOING INTO SUBROUTINE *BDUNSM*, AND THE SOLUTION WHEN RETURNS
GC........SOLUTION FROM THE CURRENT ITERATION
GF........CONVERGED SOLUTION FOR THE PREVIOUS TIME
GP........SOLUTION FROM THE PREVIOUS ITERATION
IEL.......ELEMENT TYPE: IEL=1, LINEAR; IEL=2, QUADRATIC
IMESH......INDICATOR FOR MESH GENERATION (ONLY FOR RECTANGULAR DOMAINS): IMESH=0, NOT GENERATE; IMESH=1, GENERATE
ISBC......ARRAY OF NODE NUMBER AND DEGREE OF FREEDOM THAT IS SPECIFIED: ISBC(I,1)=NODE NUMBER
ISBC(I,2)=DEGREE OF FREEDOM
ISBF......ARRAY OF NODE NUMBER AND D.O.F. OF SPECI. 'FORCES'
ITEM.......INDICATOR FOR UNSTEADY (ITEM=1) AND STEADY (ITEM=0) SOLUTION
ITMAX......MAXIMUM NUMBER OF ITERATIONS ALLOWED FOR CONVERGENCE IN THE NONLINEAR (N-S EQUATIONS) ANALYSIS; IT ALSO SERVES AS AN INDICATOR FOR LINEAR (ITMAX=1) ANALYSIS
NBW.......FULL BAND WIDTH OF 'GSTIF' FOR VELOCITIES
NCMAX......COLUMN DIMENSION OF GSTIF IN THE DIMENSION STATEMENT
NDF.......DEGREES OF FREEDOM AT EACH NODE (RH0,U,V,W,E)
NEM.......NUMBER OF ELEMENTS IN THE MESH
NEQ.......NUMBER OF EQUATIONS IN THE MODEL (=NNM)
NBHMA.....HALF BAND WIDTH OF 'GSTIF' FOR VELOCITIES
NNM.......NUMBER OF NODES IN THE MESH
NOD.......CONNECTIVITY MATRIX
NOSTRS....INDICATOR FOR THE STRESS CALCULATION: NOSTRS=0, DO NOT COMPUTE STRESSES; NOSTRS=1, COMPUTE STRESSES
NPRNT.....INDICATOR FOR PRINTING (NPRNT=1) OF ELEMNT SIFFNESS MATRIX (NPRNT=0, DO NOT PRINT)
NRENLDS...NUMBER OF REYNALDS NUMBERS FOR WHICH THE SOLUTION IS TO BE CALCULATED (FOR NONLINEAR CASE ONLY)
NSBC......NUMBER OF SPECIFIED BOUNDARY CONDITIONS (ESS. B.C.)
DIMENSION REQUIREMENTS

THE DIMENSIONS OF ARRAYS 'GSTIF', 'GF', 'GC', 'GP', 'NOD', 'ISBC', 'VSBC', 'DRE', 'X', 'Y', 'DX' AND 'DY' SHOULD BE
SUCH THAT THEY MEET THE REQUIREMENTS OF PROBLEM BEING SOLVED.

THE DIMENSION REQUIREMENTS OF THE ARRAYS ARE AS FOLLOWS:

GSTIF(NRMAX,NCMAX), GF(NRMAX,5) GC(NRMAX,5), GP(NRMAX,5)

NRMAX.GE.NEQ AND NCMAX.GE.NBW
NOD(N,I) WITH N.GE.NEM AND I.EQ.8
ISBC(I,J), VSBC(I) WITH I.GE.NSB AND J.EQ.2
ISBF(I,J), VSBF(I) WITH I.GE.NSBF AND J.EQ.2
DRE(I) WITH I.GE.NRNL; X(I), Y(I) AND Z(I) WITH I.GE.NNM
DX(I) WITH I.GE.IEL/NX+1; DY(J) WITH J.GE.IEL/NY+1, ETC.

SUBROUTINES USED

MESH3D.......GENERATES NX BY NY BY NZ MESH FOR PRISMATIC DOMAINS
BDUNSM.......SOLVES UNSYMMETRIC SYSTEM OF BANDED EQUATIONS
INVDOT.......COMPUTES THE INVERSE OF JACOBIAN & ITS DETERMINANT
MATMUL.......GIVES THE PRODUCT OF TWO MATRICES
SHAPE.......EVALUATES THE SHAPE FUNCTIONS AND THEIR DERIVATIVES
STF3D.......CALCULATES THE ELEMENT STIFFNESS MATRIX
STRS3D.......CALCULATES STRESSES AND PRESSURE IN EACH ELEMENT

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION GSTIF(500,200),GF(500,5),GP(500,5),GC(500,5),TITLE(20),
* ISBC(160,2),VSBC(160),ISBF(100,2),VSBF(100),IBDS(5),
* DRE(10),IBDF(5)
COMMON/STF/ELXYZ(8,3),STIF(8,8),ELF(8),V(8,5),VO(8,5)
COMMON/MSH/X(360),Y(360),Z(360),NOD(224,8),DX(10),DY(10),DZ(10)
DATA NRMAX,NCMAX,NDF/500,200,5/

PREPROCESSOR OF THE PROGRAM

INPUT DATA TO THE PROGRAM
READ 600, TITLE
READ 610, AMU,CV,R
READ 620, IEL,NPE,IMESH,ITEM,ITMAX,NPRNT,NOSTR
IF(IMESH.EQ.1)GOTO 30

IF THE DOMAIN IS NONRECTANGULAR, READ THE MESH INFORMATION

DO 20 N=1,NEM
20 READ 620, (NOD(N,I),I=1,NPE)
    READ 610, (X(I),Y(I),Z(I),I=1,NNM)
GOTO 40

IF THE DOMAIN IS RECTANGULAR, READ THE NUMBER OF ELEMENTS AND
COORDINATES OF THE NODES ALONG THE COORDINATE LINES (WHICH ARE
ASSUMED TO BE PARALLEL TO THE SIDES OF THE RECTANGULAR DOMAIN)

30 READ 620, NX,NY,NZ
    NXI=IEL*NX+1
    NYI=IEL*NY+1
    NZI=IEL*NZ+1
    READ 610, (DX(I),I=1,NXI)
    READ 610, (DY(I),I=1,NYI)
    READ 610, (DZ(I),I=1,NZI)
    READ 620, (NOD(1,I),I=1,NPE)
    CALL MESH3D(NX,NY,NZ,NPE,NMM,NEM)

40 NEQ=NMM

READ THE NUMBER OF SPECIFIED VELOCITIES (NSBC), THE NODE NUMBERS
(ISBC(I,1)) AND THE DEGREE OF FREEDOM (ISBC(I,2)) SPECIFIED AT
THE NODE AND THE SPECIFIED VALUES (VSBC) OF THE VELOCITIES.
SIMILARLY, READ THE SPECIFIED Boundary 'FORCES' AT THE NODES AND
SPECIFIED BOUNDARY TEMPERATURES IN CASE OF COUPLED CONVECTION.

READ 620, NSBC
READ 620, (ISBC(I,J),J=1,2),I=1,NSBC
READ 610, (VSBC(I),I=1,NSBC)
READ 620, NSBF
IF(NSBF.EQ.0)GOTO 45
READ 620, (ISBF(I,J),J=1,2),I=1,NSBF
READ 610, (VSBF(I),I=1,NSBF)

FOR NONLINEAR ANALYSIS (I.E., THE SOLUTION OF THE NAVIER-STOKES
EQUATIONS), READ THE NUMBER OF INCREMENTS OF THE REYNOLDS NUMBER
(NRENLD), THE ARRAY OF THE INCREMENTS (DRE(I)), THE ALLOWABLE
PERCENTAGE OF ERROR (TLR) BETWEEN THE VELOCITY VECTORS OF TWO
CONSECUTIVE ITERATIONS, AND THE ACCELERATION PARAMETER (BETA).

45 NRENLD=1
IF(ITMAX.LE.1)GOTO 50
READ 620, NRENLD
READ 610, (DRE(I),I=1,NRENLD)
READ 610, TLR,BETA

READ THE NUMBER OF TIME STEPS (NTIME), THE TIME STEP (DT) AND
THE PARAMETER (THETA) IN THE TIME-APPROXIMATION FOR UNSTEADY
CASE. ZERO INITIAL CONDITIONS ON THE VELOCITY ARE ASSUMED.
50 IF(ITEM.EQ.0)GOTO 60
READ 620, NTIME
READ 610, DT, THETA, EPS
60 CONTINUE

END OF THE INPUT DATA TO PROGRAM

COMPUTE THE HALF BAND WIDTH (NHBW) OF GLOBAL COEFFICIENT MATRIX

NHBW=0
DO 70 N=1, NEM
DO 70 I=1, NPE
DO 70 J=1, NPE
NW=(IABS(NOD(N, I)) - NOD(N, J)) + 1
70 IF (NHBW .LT. NW) NHBW = NW
NHBW=2*NHBW

PRINT THE INPUT DATA TO THE PROGRAM

PRINT 900
PRINT 600, TITLE
IF(ITEM.EQ.0)PRINT 870
IF(ITEM.GE.1)PRINT 880
IF(ITMAX.GT.1)PRINT 890
IF(ITEMSH.EQ.1)PRINT 910, NX, NY, NZ
PRINT 630, IEL, NDF, NEM, NNM, NBW, NSBC, NSBF
PRINT 640
DO 100 I=1, NNM
DO 80 K=1, NDF
IBDF(K)=0
80 IBDS(K)=0
DO 90 J=1, NSBC
NODE=ISBC(J, 1)
NBC=ISBC(J, 2)
IF(NODE.NE.1)GOTO 90
IBDS(NBC)=NBC
90 CONTINUE
IF(NSBF.EQ.0)GOTO 100
DO 95 J=1, NSBF
NODE=ISBF(J, 1)
NBF=ISBF(J, 2)
IF(NODE.NE.1)GOTO 95
IBDF(NBF)=NBF
95 CONTINUE
100 PRINT 650, I, X(I), Y(I), Z(I), (IBDS(K), K=1, NDF), (IBDF(K), K=1, NDF)
PRINT 660
DO 110 I=1, NEM
110 PRINT 625, I, NOD(I, J), J=1, NPE
IF(ITEM.GE.1)PRINT 920, NTIME, DT, THETA, EPS
IF(ITMAX.GT.1)PRINT 930, NRENLD, TLR, BETA

SOME BASIC CHECKS ON THE DATA

IF(IAM.EQ.0)GOTO 570
IF(NSBC.EQ.0)GOTO 560
IF(NEQ.GT.NRMAX)GOTO 580
IF(THAW.GT.NCMAX)GOTO 590
IF(IAM.EQ.1 .AND. NPE.NE.8)GOTO 550
--- PROCESSOR UNIT OF THE PROGRAM ---

LOOP ON THE REYNOLDS NUMBER BEGINS

NCOUNT=0
NRE=0
PRINT 670, RENLDS
130 NRE=NRE+1
   AMU=1.0/RENLDS
   IF(NRE.GT.NRENLD)GOTO 460

TIME LOOP FOR UNSTEADY CASE BEGINS

140 TIME=0.0
   NT=0
   INITIALIZE THE SOLUTION VECTORS: GSTIF(I,NBW), GFI(I,5) & GC(I,5)
   DO 150 I=1,NEQ
      GSTIF(I,NBW)=0.0
   DO 150 J=1,NDF
      GC(I,J)=0.0
   GFI(I,J)=0.0
160 IF(ITM.EQ.0)GOTO 175
170 NT=NT+1
    TIME=TIME+DT
    PRINT 680, TIME
    IF(NT.GT.NTIME)GOTO 450
175 NCOUNT=NCOUNT+1
   ITER=0

COUNTER ON THE ITERATIVE CYCLE ON THE NONLINEARITY BEGINS HERE

180 ITER=ITER+1
   IF (ITER.GT.ITMAX)GOTO 520
   DO 200 I=1,NEQ
      GSTIF(I,J)=0.0
   DO 200 J=1,NDF
   DO 350 NDIF=1,NDF
   DO 250 I=1,NPE
      VCI(I,J)=0.0
250 DO 280 N=1,NEM
220 DO 280 I=1,NPE
      NI=MOD(N,I)
      ELX(I,1)=X(NI)
      ELY(I,2)=Y(NI)
      ELZ(I,3)=Z(NI)
      IF(ITM.EQ.0)GOTO 210
VO(I,NDOF)=GF(NI,NDOF)
210 IF(CITMAX.LE.1)GOTO 220
VO(I,NDOF)=BETA*GP(NI,NDOF)+(1.0-BETA)*GC(NI,NDOF)
220 CONTINUE

C

IT=0
CALL FLUX3D(NPE,ELXYZ,V,AMU,IEL,CV,R,ELF)
CALL STF3D(IEL,NPE,AMU,IT,THETA,ITEM,DT)

PRINT (IF NPRNT=1) THE ELEMENT COEFFICIENT MATRIX: STIF(I,J)

IF(NPRNT.EQ.0)GOTO 240
IF(N.GT.1)GOTO 240
IF(ITER.GT.1)GOTO 240
IF(NTIME.GT.1)GOTO 240
PRINT 700
DO 250 I=1,NN
230 PRINT 610, (STIF(I,J),J=1,NN)
240 CONTINUE

ASSEMBLY OF ELEMENT MATRICES INTO GLOBAL MATRICES IN BANDED FORM

DO 260 I=1,NPE
NR= NOD(N,I)
GSTIF(NR,NBW)=GSTIF(NR,NBW)+ELF(I)
DO 260 J=1,NPE
NCL= NOD(N,J)
NC=NCL-NR+NHBW+1
IF (NC) 260,260,250
250 GSTIF(NR,NC)=GSTIF(NR,NC)+STIF(I,J)
260 CONTINUE

MODIFY THE 'FORCE' VECTOR TO INCLUDE SPECIFIED BOUNDARY VALUES

IF(NSOF.EQ.0)GOTO 300
DO 290 I=1,NPF
IF(ISOF(I,2).NE.NDOF)GOTO 290
NF=ISOF(I,1)
VF=VSOF(I)
GSTIF(NF,NBW)=GSTIF(NF,NBW)+VF
290 CONTINUE

IMPOSITION OF THE SPECIFIED BOUNDARY CONDITIONS ON VELOCITIES

DO 320 I=1,NSBC
IF(ISBC(I,2).NE.NDOF)GOTO 320
DO 310 J=1,NBW
300 GSTIF(I,E,J)=0.0
GSTIF(I,E,NBW)=1.0
GSTIF(I,E,NBW)=VSBC(I)
310 CONTINUE

SOLUTION OF THE ASSEMBLED EQUATIONS FOR THE VELOCITIES USING A
BANDED EQUATION SOLVER. THE SOLUTION IS STORED IN 'GSTIF(I,NBW)'

CALL BDUNSM(GSTIF,NRMAX,NCMAX,NEQ,NHBW,IER)

DO 330 I=1,NEQ
GP(I,NDOF)=GC(I,NDOF)
330 GC(I,NDOF)=GSTIF(I,NBH)
350 CONTINUE
NFLAG=0
IF(ITMAX.LE.1)GOTO 420
IF(NCOUNT.LE.1 .AND. ITER.LE.1)GOTO 420

CHECK FOR CONVERGENCE OF VELOCITIES FOR A GIVEN REYNALDS NUMBER

NFLAG=1
DO 400 NDOF=1,NDF
ERR=0.0
DNORM=0.0
DO 380 I=1,NEQ
DNORM=DNORM+GC(I)**2
380 ERR=ERR+(GP(I)-GC(I))**2
ERROR=DSQRT(ERR/DNORM)
IF (ERROR.GT.1L0)GOTO 180
400 PRINT 760, RENLDS,ITER,ERROR

PRINT THE LINEAR OR CONVERGED NONLINEAR SOLUTION FOR EACH NODE

420 PRINT 710
DO 430 I=1,NNM
430 PRINT 750, I,GC(I,1),GC(I,2),GC(I,3),GC(I,4),GC(I,5)

IF(ITMAX.LE.1)GOTO 440
IF(NFLAG.EQ.0)GOTO 180
440 IF(ITEM.EQ.0)GOTO 460

CHECK TO SEE IF THE SOLUTION HAS REACHED THE STEADY STATE

DIFFT=0.0
DNRM=0.0
DO 450 I=1,NEQ
DNRM=DNRM+GF(I)**2
DIFFT=DIFFT+(GF(I)-GC(I))**2
IF(CONV.EQ.0)GOTO 450
450 GF(I)=GC(I)
DIFF=1.0
IF(N.TE.1)GOTO 460
DIFF=DSQRT(DIFF/DRM)

POSTPROCESSOR PART OF THE PROGRAM

CALL SUBROUTINE 'STRS3D' TO CALCULATE STRESSES AND PRESSURE

460 IF(NOSTRS.EQ.0)GOTO 490
PRINT 740
PRINT 750
PRINT 740
DO 480 N=1,NEM
DO 470 I=1,NPE
NI=NOC(N,I)
480 J=1,NDF
465 VI(I,J) = GC(NI,J)
ELXYZ(I,1)=X(NI)
ELXYZ(I,2)=Y(NI)
470 ELXYZ(I,3)=Z(NI)
480 CALL STRES3D(NPE,ELXYZ,V,AMU,IEN,CV,R)
490 PRINT 740
C
IF(Item, EQ.0)GOTO 500
IF(DIFF.LT.EPS)GOTO 530
GOTO 170
500 IF(ITEMX,LE.1)GOTO 510
RENLDS=RENLDS+DRE(NRE)
IF(RENLDS.GT.1)GOTO 130
510 CONTINUE
GOTO 599
520 PRINT 800, ITER, ERROR
GOTO 599
530 PRINT 790, TIME
GOTO 599
550 PRINT 860
GOTO 599
560 PRINT 850
GOTO 599
570 PRINT 840
GOTO 599
580 PRINT 810
GOTO 599
590 PRINT 820
GOTO 599
599 STOP

------------------------------- FORMATS -------------------------------
600 FORMAT (20A4)
610 FORMAT (8F10.5)
620 FORMAT (16I5)
625 FORMAT (15,5X,8I5)
630 FORMAT (/5X,'ELEMENT TYPE (1, LINEAR; 2, QUADRATIC) =',I5,

* /5X,'NUMBER OF DEGREES OF FREEDOM PER NODE =',I5,
* /5X,'NUMBER OF ELEMENTS IN THE MESH =',I5,
* /5X,'NUMBER OF NODES IN THE MESH =',I5,
* /5X,'FULL BAND WIDTH OF THE ASSEMBLED MATRIX =',I5,
* /5X,'NUMBER OF SPECIFIED PRIMARY UNKNOWNS =',I5,
* /5X,'NUMBER OF SPECIFIED FLUXES =',I5)
640 FORMAT (/2X,'NODE',10X,'X',16X,'Y',16X,'Z',16X,'SPEC. VEL.',5X,
* 'SPEC. FORCES',2X,'SPEC. TEMP.',/,70X,'O INDICATES UNSPEC.',/
650 FORMAT (15,5X,12.5),3X,3I5,2X,3I5,2X,15)
660 FORMAT (/5X,'ELEMENT STIFFNESS MATRIX =',/
670 FORMAT (/5X,'R E Y N A L D S N U M B E R =',E12.4)
680 FORMAT (/5X,'T I M E =',E12.4,/
700 FORMAT (/5X,'ELEMENT STIFFNESS MATRIX =',/
710 FORMAT (/5X,'NODE',2X,'DENSITY',5X,'X-VELOCITY',5X,'Y-VELOCITY',
* 'Z-VELOCITY',5X,'TEMPERATURE',/
730 FORMAT (5X,15,5X,2X,E13.5))
760 FORMAT (5X,120(''))
SUBROUTINE STF3D(IIF,NSC) *FIF* IS READ AS ZERO. IT
*SHOULD BE EITHER 1 OR 2."
850 FORMAT (5X, "INPUT ERROR": THE VALUE OF NSC IS EQUAL TO ZERO")
860 FORMAT (5X, "INPUT ERROR": VALUES OF IEL AND NPE DO NOT MATCH.
*IF IEL=1 THEN NPE=4, AND IF IEL=2 THEN NPE=8 OR 9")
870 FORMAT (5X, "STEADY-STATE SOLUTION")
880 FORMAT (5X, "UNSTEADY SOLUTION")
890 FORMAT (5X, "ONLINEAR ANALYSIS")
900 FORMAT (5X, "PROGRAM FLOPEN3D (DEVELOPED
*BY J. N. REDDY)", /,,5X,75(I1,-",/)
910 FORMAT (5X, "INPUT ERROR": THE VALUE OF NSC IS EQUAL TO ZERO")
920 FORMAT (5X, "INPUT ERROR": VALUES OF IEL AND NPE DO NOT MATCH.
*IF IEL=1 THEN NPE=4, AND IF IEL=2 THEN NPE=8 OR 9")
930 FORMAT (5X, "UNSTEADY SOLUTION")
940 FORMAT (5X, "ONLINEAR ANALYSIS")
950 FORMAT (5X, "PROGRAM FLOPEN3D (DEVELOPED
*BY J. N. REDDY)", /,,5X,75(I1,-",/)
960 FORMAT (5X, "INPUT ERROR": THE VALUE OF NSC IS EQUAL TO ZERO")
970 FORMAT (5X, "INPUT ERROR": VALUES OF IEL AND NPE DO NOT MATCH.
*IF IEL=1 THEN NPE=4, AND IF IEL=2 THEN NPE=8 OR 9")
980 FORMAT (5X, "UNSTEADY SOLUTION")
990 FORMAT (5X, "ONLINEAR ANALYSIS")
END

PROGRAM 'STF3D' GENERATES ELEMENT COEFFICIENT MATRIX 'STIF' AND
SOURCE VECTOR 'ELF' FOR THE LINEAR (EIGHT-NODE) ISOPARAMETRIC
PRISM ELEMENT.

GAUSS....ARRAY OF GAUSS POINTS
GDSF....GLOBAL DERIVATIVES OF THE SHAPE FUNCTIONS
GDSF(I,J)=DERIVATIVE OF SF(J) W.R.T X(I)
WT....ARRAY OF WEIGHTS CORRESPONDING TO GAUSS POINTS
SF....ELEMENT SHAPE FUNCTIONS
STIF....ELEMENT COEFFICIENT MATRICES

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION GAUSS(4,4),W(4,4),S(8,8)
COMMON/STF/STF(SF8,STIF(STIF8,STIF8)),V8,VO8
COMMON/SHP/SF8,SHDF3D(3,8)

DATA GAUSS/4X0.0D0,-.77350270D0,.77350270D0,2X0.0D0,-.77459667D0,
*0.0D0,.77459667D0,0.0D0,-.86113631D0,-.35980104D0,0.35980104D0,
*0.86113631D0/
DATA WT/2.0D0,3X0.0D0,2X1.0D0,2X0.0D0,0.55555555D0,0.88888888D0,
*0.55555555D0,0.0D0,0.34785485D0,2X0.65214515D0,0.34785485D0/
NGP=IEL+1
NGP1=IEL

INITIALIZE THE ARRAYS, SX, SY, ETC. (FOR PENALTY TERMS)

DO 50 I = 1, NPE
   ELF(I) = 0.0
DO 50 J = 1, NPE
   SCI(J) = 0.0
   50 STIF(I,J) = 0.0
   IF(IT.EQ.1) GO TO 145

COMPUTE THE COEFFICIENT MATRICES FOR EACH VARIABLE

145 DO 300 NI = 1, NGP
   DO 300 NJ = 1, NGP
   DO 300 NK = 1, NGP
   XI = GAUSS(NI,NGP)
   ETA = GAUSS(NJ,NGP)
   ZETA = GAUSS(NK,NGP)
   CALL SHPSDF(NPE,DET,XI,ETA,ZETA,ELXYZ)
   CONST=DET**NT(NI,NGP)**NT(NJ,NGP)**NT(NK,NGP)
   RHO=0.0
   T=0.0
   V1 = 0.0
   V2 = 0.0
   V3 = 0.0
   DIV = 0.0
   DO 150 I = 1, NPE
      RHO=RHO+V(I,1)**SF(I)
      T=T+V(I,5)**SF(I)
      V1 = V1+V(I,2)**SF(I)
      V2 = V2+V(I,3)**SF(I)
      V3 = V3+V(I,4)**SF(I)
      150 DIV=DIV+V(I,2)**GDSF1(I)+V(I,3)**GDSF2(I)+V(I,4)**GDSF3(I)
   DO 200 I = 1, NPE
   160 DO 200 J = 1, NPE
      STIF(I,J) = STIF(I,J) + ((V1**GDSF1(I,J)+V2**GDSF2(I,J)+V3**GDSF3(I,J))**SF(I)**SF(J)**DIV)**CONST
   SCI(J) = SCI(J) + SF(I)**SF(J)**CONST
   200 CONTINUE
   300 CONTINUE
   IF(IT.EQ.0) RETURN
   THETA=1.0-THETA
   DO 600 I = 1, NN
   DO 600 J = 1, NN
      ELF(I) = ELF(I) + (S(I,J)-DT**THETA**STIF(I,J))**V0(J)
   600 STIF(I,J) = S(I,J) + DT**THETA**STIF(I,J)
   RETURN
END

SUBROUTINE FLUX3D(NPE,ELXYZ,V,AMU,IEL,CV,R,ELF,NDOF)

PROGRAM 'FLUX3D' COMPUTES TOTAL STRESSES, PRESSURE AND DIVERGENCE
OF THE VELOCITY FIELD AND THE FLUX VECTORS FOR EACH EQUATION.
P = PRESSURE AT THE CURRENT GAUSS POINT
DPX = DERIVATIVE OF P WITH RESPECT TO X
DPY = DERIVATIVE OF P WITH RESPECT TO Y
DPZ = DERIVATIVE OF P WITH RESPECT TO Z
DIVQ = DIVERGENCE OF THE FLUX
**DISPN = DISSIPATION, SIGMA:STRAIN RATE**

```
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION ELXYZ(8,3),V(8,5),GAUSS(4,4)
COMMON/SHP/SF(8),GDSF(3,8)
DATA GAUSS/6*0.0D0,-.57735027D0,.57735027D0,2*0.0D0,-.77459667D0,
  *0.0D0,.77459667D0,0.0D0,-.86113631D0,-.33998104D0,.33998104D0,
  *0.86113631D0/
NGP=IEL
DO 100 II=1,NGP
DO 100 JJ=1,NGP
DO 100 KK=1,NGP
XI=GAUSS(II,NGP)
ETA=GAUSS(JJ,NGP)
ZETA=GAUSS(KK,NGP)
CALL SHP3D(NPE,DET,XI,ETA,ZETA,ELXYZ)
X = 0.0
Y = 0.0
Z = 0.0
SUM1 = 0.0
SUM2 = 0.0
SUM3 = 0.0
SUM12=0.0
SUM13=0.0
SUM23=0.0
RHO=0.0

CALCULATE STRAIN-RATES

DO 40 I=1,NPE
  X = X + ELXYZ(I,1)*SF(I)
  Y = Y + ELXYZ(I,2)*SF(I)
  Z = Z + ELXYZ(I,3)*SF(I)
  RHO=RHO+WI(I,1)*SF(I)
  SUM1=SUM1+WI(I,2)*GDSF(1,I)
  SUM2=SUM2+WI(I,3)*GDSF(2,I)
  SUM3=SUM3+WI(I,4)*GDSF(3,I)
  SUM12=SUM12+WI(I,2)*GDSF(2,I)+WI(I,3)*GDSF(1,I)
  SUM13=SUM13+WI(I,2)*GDSF(3,I)+WI(I,4)*GDSF(1,I)
  SUM23=SUM23+WI(I,3)*GDSF(3,I)+WI(I,4)*GDSF(2,I)
40 CONTINUE

CALCULATE STRESSES AND PRESSURE

USE PROPER CONSTITUTIVE LAWS TO COMPUTE THE TEMPERATURE AND
PRESSURE

CALL STATE(XI,ETA,P,DPX,DPY,DPZ,DIVQ,V,CA,R)

SIGMA1 = -P+AMU*(4.0*SUM1-2.0*(SUM2+SUM3))/3.0
SIGMA2 = -P+AMU*(4.0*SUM2-2.0*(SUM1+SUM3))/3.0
SIGMA3 = -P+AMU*(4.0*SUM3-2.0*(SUM1+SUM2))/3.0
SIGMA12=AMU*SUM12
SIGMA13=AMU*SUM13
SIGMA23=AMU*SUM23
IF(NDOF.EQ.2)ELF(I)=ELF(I)+SF(I)*(-DPX+SIGMA1X+SIGMA12X+SIGMA13Z)*
  *CONST
IF(NDOF.EQ.3)ELF(I)=ELF(I)+SF(I)*(-DPY+SIGMA12X+SIGMA23X+SIGMA13Z)*
  *CONST
```
IF(NDOF.EQ.4)ELF(I)=ELF(I)+SF(I)*(-DPZ+SGMA13X+SGMA23Y+SGMA3Z)*
*     CONST
IF(NDOF.EQ.5)ELF(I)=ELF(I)+SF(I)*(-DIVQ+DISP)*CONST
50  CONTINUE
100  FORMAT (5X,10E12.4)
    RETURN
   END
SUBROUTINE SHP3D(NPE,DET, XI, ETA, ZETA, ELXYZ)

PROGRAM 'SHP3D' EVALUATES SHAPE FUNCTIONS AND THEIR DERIVATIVES
AT THE GAUSSIAN POINTS OF THE EIGHT-NODE ISOPARAMETRIC PRISMATIC
ELEMENT. SHAPE FUNCTIONS FOR HIGHER-ORDER ELEMENTS CAN BE ADDED

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION XNODE(8,3),DSF(3,8),ELXYZ(8,3),GJ(3,3),GJINV(3,3)
COMMON/SHP/SF(8),GDSF(3,8)
DATA XNODE/2*1.0D0,2*1.0D0,2*1.0D0,2*1.0D0,2*1.0D0,2*1.0D0,2*1.0D0,
*     2*1.0D0,2*1.0D0,2*1.0D0,2*1.0D0,2*1.0D0,2*1.0D0/
C
FNC(A,B,C) = 0.125*A*B*C
DO 50 I = 1, NPE
   XP=XNODE(I,1)
   YP=XNODE(I,2)
   ZP = XNODE(I,3)
   XI = 1.0 + XI*XP
   ETA = 1.0 + ETA*YP
   ZETA = 1.0 + ZETA*ZP
   SF(I) = FNC(XI,ETA,ZETA)
   DSF(1,I) = FNC(XP, ETA, ZETA)
   DSF(2,I) = FNC(XP, ETA, ZETA)
   DSF(3,I) = FNC(XP, ETA, ZETA)
50  CONTINUE
CALL MATML(DSF,3,NPE,ELXYZ,3,GJ)
CALL INVDET(GJ,GJINV,DET)
CALL MATML(GJINV,3,3,DSF,NPE,GDSF)
RETURN
END
SUBROUTINE STRS3D(NPE,ELXYZ, W, AMU, IEL, CV, R)

PROGRAM 'STRS3D' COMPUTES TOTAL STRESSES, PRESSURE AND DIVERGENCE
OF THE VELOCITY FIELD IN EACH ELEMENT (TO CHECK CONSERV. OF MASS)

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION ELXYZ(8,3),W(8,3),GAUSS(4,4)
COMMON/SHP/SF(8),GDSF(3,8)
DATA GAUSS/4*0.0D0,-7775027D0,57735027D0,2*0.0D0,-77459667D0,
*     0.0D0,77459667D0,0.0D0,-0.86113631D0,-0.33998104D0,33998104D0,
*     0.86113631D0/
LG=IEL
DO 50 II=1,NGP
DO 50 JJ=1,NGP
DO 50 KK=1,NGP
XI=GAUSS(II,NGP)
ETA=GAUSS(JJ,NGP)

ZETA=GAUSS(KK,NGP)
CALL SHP3D(NPE,DET,XI,ETA,ZETA,ELXYZ)
X = 0.0
Y = 0.0
Z = 0.0
SUM1 = 0.0
SUM2 = 0.0
SUM3 = 0.0
SUM12 = 0.0
SUM13 = 0.0
SUM23 = 0.0
RHO=0.0

CALCULATE STRAIN-RATES

DO 40 I=1,NPE
  X = X + ELXYZ(I,1)*SF(I)
  Y = Y + ELXYZ(I,2)*SF(I)
  Z = Z + ELXYZ(I,3)*SF(I)
  RHO=RHO+W(I,1)*SF(I)
  SUM1=SUM1+W(I,2)*GDSF(1,1)
  SUM2=SUM2+W(I,3)*GDSF(2,1)
  SUM3=SUM3+W(I,4)*GDSF(3,1)
  SUM12=SUM12+W(I,2)*GDSF(2,1)+W(I,3)*GDSF(1,1)
  SUM13=SUM13+W(I,1)*GDSF(3,1)+W(I,4)*GDSF(1,1)
  SUM23=SUM23+W(I,3)*GDSF(3,1)+W(I,4)*GDSF(2,1)
40

CALCULATE STRESSES AND PRESSURE

USE PROPER CONSTITUTIVE LAWS TO COMPUTE THE TEMPERATURE AND PRESSURE

T = CV*RHO
P = RHO*X T
SIGMA1 = -P+AMUX(4.0*SUM1-2.0*(SUM2+SUM3))/3.0
SIGMA2 = -P+AMUX(4.0*SUM2-2.0*(SUM1+SUM3))/3.0
SIGMA3 = -P+AMUX(4.0*SUM3-2.0*(SUM1+SUM2))/3.0
SIGMA12=AMUX*SUM12
SIGMA13=AMUX*SUM13
SIGMA23=AMUX*SUM23
PRINT 200,X,Y,Z,SIGMA1,SIGMA2,SIGMA3,SIGMA12,SIGMA13,SIGMA23,P
50 CONTINUE
100 CONTINUE
200 FORMAT (5X,10E12.4)
RETURN
END

SUBROUTINE INVDET (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(3,2),A(2,3))
B(1,2)=F(A(1,2),A(3,3),A(1,3),A(3,2))
B(1,3)=F(A(1,2),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(1,3),A(1,2))
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 MATMLT(A,M,N,B,L,C)

PROGRAM 'MATMLT' GIVES THE PRODUCT OF TWO MATRICES: 
    [C]=[A][B]
    THE PROGRAM Multiplies A(M,N) BY B(N,L) TO GIVE C(M,L)

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(M,N),B(N,L),C(M,L)
DO 10 I=1,M
    DO 10 J=1,L
    C(I,J)=0.
    DO 10 K=1,N
        10 C(I,J)=C(I,J)+A(I,K)×B(K,J)
RETURN
END

SUBROUTINE MESH3D(NEX,NEY,NEZ,NPE,NNM,NEM)

PROGRAM 'MESH3D' GENERATES MESH FOR RECTANGULAR DOMAINS USING
    THE EIGHT-NODE (LINEAR) PRISMATIC ELEMENTS. THE PROGRAM COMPUTES
    THE COORDINATES OF GLOBAL NODES X,Y, AND Z, BOOLEAN CONNECTIVITY
    MATRIX 'NOD' RELATING ELEMENT NODES TO GLOBAL NODES OF THE MESH.

IMPLICIT REAL*8 (A-H,O-Z)
COMMON/MSH/X(360),Y(360),Z(360),NOD(224,8),DX(10),DY(10),DZ(10)
NPEH=NPE/2
NEY=NEX×NEY
NEM = NEXY×NEZ
NPX=NEX+1
NPY=NEY+1
NPZ=NEZ+1
NNM = NPX×NPY×NPZ

GENERATE THE CONNECTIVITY MATRIX, 'NOD'

N=1
N1=NPX
N2=NPX
N3=NPX×NPY
NN = N3×NPZ
KK = 1
IF (NEX .EQ. 1) GOTO 30
    DO 20 I=2,NEX
        N=N+1
    20 DO 10 K=1,NPE

10 NOD(I,K)=NOD(I-1,K)+KK
20 CONTINUE
30 IF(NEZ .EQ. 1) GOTO 50
   DO 40 J=2,NEZ
   DO 40 I=1,NEX
   N=N+1
   M=M-NEZ
   DO 40 K=1,NPE
   NPIY=N2
   IF(K .EQ. 0) NPIY = N1
   NOD(N,K)=NOD(M,K)+NPIY
40 CONTINUE
50 IF(NEZ .EQ. 1) GOTO 70
   DO 60 I=2,NEZ
   DO 60 K=1,NEXY
   N=N+1
   M=M-NEXY
   DO 60 I=1,NPE
60 NOD(N,I)=NOD(M,I)+N3
70 CONTINUE

COMPUTE THE COORDINATES X, Y, AND Z OF THE GLOBAL NODES

   DO 80 K=1,NPZ
   DO 80 J=1,NPY
   DO 80 I=1,NPX
   N=1+(I-1)*KK+(J-1)*N2+(K-1)*N3
   X(N)=DX(I)
   Y(N)=DY(J)
   Z(N)=DZ(K)
80 RETURN
END

SUBROUTINE BDUNSM(A,NRMAX,NCMAX,N,ITERM,IER)

PROGRAM 'BDUNSM' IS AN EQUATION SOLVER FOR BANDED UNSYMMETRIC
SET OF EQUATIONS. THE SOLUTION IS STORED IN THE LAST COLUMN OF A

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(NRMAX,NCMAX)
CERO=1.0D-10
PARE=CERO*Q2
NBND=2*ITERM
NBMIN=NBND-1

BEGIN ELIMINATION OF THE LOWER LEFT

   DO 80 I=1,N
   IF (DABS(A(I,ITERM)) .LT. CERO) GO TO 10
   DO 10 J=1,ITERM-1,N
10   L=ITERM+1
   DO 40 J=1,JLAST
   DO 40 L=1,L
   IF (DABS(A(J,L)) .LT. PARE) GO TO 40
   B=A(J,L)
   DO 30 K=L,NBND
30   A(J,K)=A(J,K)/B
   DO 10 J=1,N
   IF (DABS(A(I,J)) .LT. CERO) GO TO 10
   IF (DABS(A(I,J)) .LT. PARE) GO TO 40
   B=A(I,J)
   DO 30 K=1,NBND
30   A(I,K)=A(I,K)/B
IF (I.EQ.N) GO TO 90
40 CONTINUE
    L=0
    JFIRST=I+1
    IF (JLAST.LE.I) GO TO 80
    DO 70 J=JFIRST,JLAST
        L=L+1
        JL=J-L
        ITL=ITERM-L
        IF (DABS(A(J,ITL)) .LT. PARE) GO TO 70
        DO 50 K=ITERM,NBM
            KL=K-L
            A(J,KL) = A(JL,K) - A(J,KL)
            A(J,NBND)= A(JL,NBND)-A(J,NBND)
        IF (I.GE.N-ITERM-1) GO TO 70
        DO 60 K=1,L
            NBK=NBND-K
            60 A(J,NBK) = - A(J,NBK)
    70 CONTINUE
80 CONTINUE
90 L=ITERM-1
    DO 100 I=2,N
        NII=N+1-I
        DO 100 J=1,L
            NIJ=NII+J
            ITJ=ITERM+J
            IF (N+I-I+J.GT.N) GO TO 100
            A(NII,ITJ) = A(NII,NBND) - A(NIIJ,NBND) * A(NII,ITJ)
    100 CONTINUE
100 IER=0
110 PRINT 130
120 FORMAT (10X,'THE EQUATION NUMBER AND THE COEFFICIENT =',I5,E13.4)
130 FORMAT (5X,'COMPUTATION STOPPED IN BDUNSM BECAUSE ZERO APPEARED ON X THE MAIN DIAGONAL. *** CHECK YOUR MATRIX ***')
END
## THE UNIVERSITY OF TENNESSEE SPACE INSTITUTE

**CONTRACT NO.:** NASA NIAS-36555  
**UT ACCT NO.:** R02-400450  
**PRINCIPAL INVEST: DR. K.C. REDDY**  
**EFFECTIVE DATE: 03/13/86 - 03/31/87**

<table>
<thead>
<tr>
<th>MONTH</th>
<th>PERSONNEL SERVICES</th>
<th>FRINGE BENEFITS</th>
<th>TRAVEL</th>
<th>COMMUNICATIONS</th>
<th>COMPUTER USAGE</th>
<th>SUPPLIES &amp; OTHER</th>
<th>EQUIPMENT</th>
<th>OVERHEAD</th>
<th>TOTAL EXPDITURES</th>
<th>CHARGES OUTSTANDING</th>
<th>BUDGET ADDITIONS</th>
<th>FREE BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUDGET</td>
<td>$23,930</td>
<td>$1,337</td>
<td>$4,000</td>
<td>$400</td>
<td>$2,000</td>
<td>$0</td>
<td>$3,000</td>
<td>$15,327</td>
<td>$49,994</td>
<td></td>
<td></td>
<td>$49,994</td>
</tr>
<tr>
<td>BAL FWRD</td>
<td>$8,156</td>
<td>$1,442</td>
<td>$833</td>
<td>$0</td>
<td>$0</td>
<td>$34</td>
<td>$0</td>
<td>$5,390</td>
<td>$15,855</td>
<td></td>
<td></td>
<td>$34,139</td>
</tr>
<tr>
<td>AUG 86</td>
<td>$1,867</td>
<td>$276</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$1,104</td>
<td>$3,247</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO DATE</td>
<td>$10,023</td>
<td>$1,718</td>
<td>$833</td>
<td>$0</td>
<td>$0</td>
<td>$34</td>
<td>$0</td>
<td>$6,493</td>
<td>$19,102</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEPT 86</td>
<td>$1,287</td>
<td>$74</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$701</td>
<td>$2,063</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO DATE</td>
<td>$11,310</td>
<td>$1,793</td>
<td>$833</td>
<td>$0</td>
<td>$0</td>
<td>$34</td>
<td>$0</td>
<td>$7,195</td>
<td>$21,164</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCT 86</td>
<td>$1,890</td>
<td>$251</td>
<td>$48</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$1,127</td>
<td>$3,316</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO DATE</td>
<td>$13,200</td>
<td>$2,044</td>
<td>$880</td>
<td>$0</td>
<td>$0</td>
<td>$34</td>
<td>$0</td>
<td>$8,322</td>
<td>$24,480</td>
<td></td>
<td></td>
<td>$25,514</td>
</tr>
</tbody>
</table>