PROGRAMMER’S AND USER’S MANUAL
for the

FDNS-RFV/PVM CFD CODE

Version 2000

Gary C. Cheng

Mechanical Engineering Department
University of Alabama at Birmingham
BEC 356B
1530 3rd Ave. S.
Birmingham, AL 35294-4461

TEL: 205-934-2038
FAX: 205-975-7217

e-mail: gcheng@uab.edu

September, 2002
FOREWORD

The FDNS-RFV/PVM code is a general purpose CFD solver developed by SECA, Inc. of Huntsville, AL. The code is mature and versatile having been used successfully to simulate a large number of flowfields related to aerospace propulsion systems for NASA/MSFC, Lockheed Martin Corp., and the Air Force. Due to its generality, it is not simple to use. However, those interested in its use are invited to contact Dr. Gary Cheng by phone at (205) 934-2038, FAX at (205) 934-8485, or e-mail at gcheng@uab.edu. Although this code is currently a practical computational tool, future releases are planned to increase its generality, simplify its use, and improve its computational efficiency.
## TABLE OF CONTENTS

1. INTRODUCTION 1

2. NUMERICAL METHOD 2

   2.1 Governing Equation 2

      2.1.1 Homogeneous Fluid Model 5
      2.1.2 Pressure Correction Equation 6
      2.1.3 Wall Function Model 7
      2.1.4 Compressible Turbulence Models 7
      2.1.5 Chemistry Models 8
      2.1.6 Particulate Multi-phase Model 11

   2.2 Numerical Scheme 13

      2.2.1 Adaptive Dissipation Scheme 13
      2.2.2 Viscous Terms Discretizations 15
      2.2.3 Time-Marching Scheme 16

   2.3 Boundary Conditions 16

      2.3.1 Inlet Flow Boundaries 16
      2.3.2 Exit Flow Boundaries 17
      2.3.3 Symmetry Boundaries 17
      2.3.4 Zonal Interface Boundaries 17
      2.3.5 Singularity Boundaries 18
      2.3.6 Wall Boundaries 18

   2.4 Initial Conditions 19
2.4.1 Reference Conditions

2.4.2 Normalization of Flow Variables

2.5 References

3. FDNS-RFV CODE FEATURES

3.1 Code Structure

3.2 Input Data Definition

3.3 Pre-/Post-Processor for Parallel Computing

3.4 User Defined Run-Time Modifications

  3.4.1 Main Program Includes (fmain01 & fmain02)

  3.4.2 Example Subroutine Includes (fexmp01)

3.5 Restart/Output Files (in Main and DATAIO)

4. APPENDIX A

5. APPENDIX B

6. APPENDIX C

7. APPENDIX D
1. INTRODUCTION

This user’s and programmer’s manual is centered around the description of a Real Fluids Version of a Finite-Difference Navier-Stokes, FDNS-RFV/PVM, computational fluid dynamics (CFD) code for 2-D/3-D flow field simulations and analyses. The FDNS-RFV/PVM code, a parallelization version of the FDNS-RFV code with the PVM libraries, was developed for engineering analyses of general fluid dynamics problems. A wide range of validation cases has been presented through reports, papers and technical presentations.

The current version of the FDNS-RFV/PVM code can be used to solve the 2-D planar, axisymmetric or 3-D forms of the Navier-Stokes equations and other scalar transport equations using primitive variables and general multi-zone multi-block curvilinear coordinates. Compressible or incompressible, laminar or turbulent flow problems are solved using a pressure based predictor/multi-corrector solution algorithm. Perfect gas, real fluid and reacting flow, with and without solid fuel burning surface, options are provided in the current version. In order to analyze the effect of the phase change in the spray combustion flows, real-fluid thermal and caloric equations of state (EOS) were developed and interfacial heat and mass transfer effects were neglected for the very high pressure (near or above critical pressure) in existing rocket engine combustion chambers. The HBMS equations of state (Ref. 1-2) were selected for this purpose. These equations are not only of sufficiently high order that properties are accurately predicted for a wide range of conditions, but component submodels may be easily modified. In this instance, the vapor pressure curve and the liquid phase density correlations were improved over the original HBMS formulation. Multicomponent mixture properties were calculated by adding partial specific volumes or pressure. The partial volume methodology is essential to provide accurate and robust predictions when a small amount of multi-component vapor and a large amount of liquid are present at the same point. The properties routines also include correlations for the transport properties. This was convenient since much of the methodology was similar. Furthermore, this real-fluid property submodel can be ported into other CFD code or used as a stand-alone program. The real fluids properties automatically reduce to ideal gas properties at low pressures, so that only one CFD code is needed.

The fundamental formulation of the FDNS-RFV/PVM methodology is described in Section 2 of this manual. The structure the FDNS-RFV/PVM code as well as functions of each subroutines are described in Section 3. The user’s guide for the control cards, which comprise part of the input to the FDNS-RFV/PVM code (through Fortran unit 11), is also included in Section 3. A sample input file for an example problem of the tri-propellant shear coaxial injector flow is listed in APPENDIX A. Since the FDNS-RFV/PVM code is the general flow solver, pre-processors (such as a grid generator and a flow initializer) are required to provide the code with the configuration and initial conditions for a specific flow problem. In addition, visualization tools (such as PLOT3D™ and TECPLOT™) are needed to post-process the numerical results computed by the code.
2. NUMERICAL METHOD

The FDNS-RFV/PVM\textsuperscript{3-10} code is used to solve a set of nonlinear and coupled transport equations, e.g. the Navier-Stokes equations, energy equation, two-equation turbulence models and chemical species continuity equations in non-dimensional form. Finite difference approximations are employed to discretize the transport equations on a non-staggered grid system. High order upwind, total variation diminishing (TVD), or second-order central differencing schemes plus adaptive second-order and fourth-order dissipation terms are used to approximate the convection term of the transport equations such that computed supersonic flows including shock structure will be stabilized. Second-order central differencing schemes are used for the viscous and source terms of the governing equations. Vectorized point implicit, conjugate gradient, and generalized minimal residual matrix solvers\textsuperscript{11} (GMRES) are optionally employed to insure a stable, accurate and fast convergence rate. Multi-block, multi-zone options are included in the FDNS-RFV/PVM code so that problems with complex geometries can be analyzed efficiently. A pressure based predictor/multi-corrector solution procedure is employed in FDNS-RFV/PVM for both compressible and incompressible flow problems to ensure velocity-pressure coupling and, for incompressible flows, divergence-free flowfield solutions at the end of each time marching step. A time-centered Crank-Nicholson or Euler implicit time-marching scheme is used for the temporal discretization for time accurate solutions. For steady state problems, an implicit Euler time or explicit marching scheme is recommended. The selection of type of time marching scheme can be controlled through input data. Options to account for either finite rate or equilibrium chemistry are provided in the code to compute reacting flow problems. In addition, a homogeneous fluid model is incorporated in the code to account for the effects of phase change in flows such as spray combustion. Lagrangian-Euler methodology is also employed to account for the effects of solid particles/gas flow interactions in the multi-phase flow problems.

In the current version of the FDNS-RFV/PVM code, two-equation, incompressible or compressible, standard or extended\textsuperscript{12}, $k$-$\epsilon$ turbulence models with a wall function or direct integration to the wall (low-Reynolds number turbulence model) options are included. Turbulence model options are selected through input data. Chemical kinetics and species thermodynamics data\textsuperscript{13} are required in the input data file.

2.1 GOVERNING EQUATIONS

The basic equations employed in the current version of the FDNS-RFV/PVM code to describe general 2-D/3-D flowfield are the curvilinear-transformed, multi-component conservation equations. A generalized form of these equations based on eddy viscosity turbulence mixing concepts is written as:

$$\frac{1}{J} \frac{\partial}{\partial t} (\rho \, q) = - \frac{\partial}{\partial \xi_i} (\rho \, U_i \, q) + \frac{\partial}{\partial \xi_i} \left( \mu_i \, G_{ij} \frac{\partial q}{\partial \xi_j} \right) + S_q$$

where $q = 1, u, v, w, h, k, \epsilon$ and $a_i$ for the continuity, momentum, energy (static enthalpy), turbulence model and chemical species transport equations respectively. $J$, $U_i$ and $G_{ij}$ represent the Jacobian of coordinate transformation, volume-weighted contravariant velocities and diffusion metrics respectively. They are defined as:
where \( u_j \) is the velocity vector and indicial notation is used. The Jacobian is given as:

\[
J = \frac{\partial (\xi, \eta, \zeta)}{\partial (x, y, z)} ; \quad U_i = \frac{u_j}{J} \frac{\partial \xi_i}{\partial x_j} ; \quad G_{ij} = \frac{1}{J} \frac{\partial \xi_i}{\partial x_k} \frac{\partial \xi_j}{\partial x_k}
\]

These transformation metrics can be calculated using central differences for the interior points of the mesh system. One-sided differencing can be applied to boundary grid points. For 2-D planar or axisymmetric flow problems, the Jacobian is replaced by

\[
J = y^{(1AX - 1)} z^{(2AX - 2)} \frac{\partial (\xi, \eta)}{\partial (x, y)}
\]

where IAX is one of the input parameters. IAX = 1 is used for 2-D planar flow problems and IAX = 2 is specified for axisymmetric flow problems. For axisymmetric problems, the radial direction is assumed to be along the positive y-axis (i.e. always use positive y for axisymmetric problems). The z-coordinate here simulates the effects of 3-D volume changes (i.e. thickness in the third direction) with a default value of unity. The effective viscosity in the governing equations represents a sum of the fluid viscosity, \( \mu \), and the turbulent eddy viscosity, \( \mu_t \), then re-scaled by a turbulence Prandtl or Schmidt number, \( \sigma_q \). That is

\[
\mu_e = \frac{\mu + \mu_t}{\sigma_q} ; \quad \text{where} \quad \mu_t = \rho C_\mu \frac{k^2}{\varepsilon}
\]

\( C_\mu = 0.09 \) is a turbulence modeling constant. The turbulence eddy viscosity is re-scaled by wall damping functions when low-Reynolds number turbulence models are used. Source terms of the transport equations, \( S_q \), are given by

\[
S_q = \frac{1}{J} \left\{ \left( \frac{\partial p}{\partial x_i} + \frac{\partial \mu_e}{\partial x_j} \frac{\partial u_j}{\partial x_i} + \frac{2 \lambda_c}{3} \frac{\partial \mu_e}{\partial x_i} \frac{\partial u_j}{\partial x_j} + \frac{2 \lambda_c}{3} \frac{\partial \mu_e}{\partial x_i} \left( \frac{\partial u_j}{\partial x_j} \right) + (1 - \delta_{jk}) \frac{\partial \mu_e}{\partial \xi_j} \left( J G_{ikj} \mu_e \frac{\partial u_i}{\partial \xi_k} \right) \right) + F_i \right\}
\]

\[
\frac{Dp}{Dt} + \Phi + Q_l
\]

\[
\rho \left( \frac{Dp}{Dt} + \Phi + Q_l \right)
\]

\[
\rho \frac{\varepsilon}{k} \left[ \left( C_1 + C_3 \frac{P_r \varepsilon}{\varepsilon} \right) P_r - C_2 \varepsilon \right]
\]

\[
\omega_k ; \quad k = 1, 2, \ldots, N
\]
where \( N \) is the number of species, and the subscript \( i = 1, 2, \) and \( 3 \) for \( u, v, \) and \( w \) momentum equations, respectively. \( F_i \) includes various body forces such as the gravity and the centrifugal force in \( i \)-direction. \( \delta_{jk} \) \((j,k=1,2,3)\) is the Kronecker delta function, and \( \lambda_c \) is equal to unity for compressible flows and 0 for incompressible flows. Values of \( \sigma_q \) and turbulence modeling constants are given in Tables 2.1 and 2.2. \( P_t \) stands for the turbulence kinetic energy production rate which is defined in Cartesian coordinate as:

\[
P_t = \frac{\mu}{\rho} \left[ 2 \left( u_x^2 + v_y^2 + w_z^2 \right) + (v_x + u_y)^2 + (w_y + v_z)^2 + (u_z + w_x)^2 - \frac{2}{3} \lambda_c \left( u_x + v_y + w_z \right)^2 \right]
\]

| Table 2.1 Values of \( \sigma_q \) for the Transport Equations |
|-----------------|---------------|---------------|
|                 | Laminar Flow  | Turbulent Flow|
| Momentum Equations | 1.00           | 1.00           |
| Energy Equation   | 0.72           | 0.90           |
| \( k \)-Eq. (Standard Model) | -----     | 1.00           |
| \( \varepsilon \)-Eq. (Standard Model) | -----     | 1.30           |
| \( k \)-Eq. (Extended Model) | -----     | 0.8927         |
| \( \varepsilon \)-Eq. (Extended Model) | -----     | 1.15           |
| Species Equations | 1.00           | 0.90           |

| Table 2.2 Turbulence Modeling Constants |
|-----------------|---------------|---------------|
| Constants       | Standard Model| Extended Model|
| \( C_\mu \)     | 0.09          | 0.09          |
| \( C_1 \)       | 1.43          | 1.15          |
| \( C_2 \)       | 1.92          | 1.90          |
| \( C_3 \)       | 0.00          | 0.25          |

For compressible flows, better predictions (especially for shear layers and/or separated flows) can be obtained by employing compressibility corrections to the above turbulence models. The compressibility corrected turbulence models included in the FDNS-RFV/PVM code are discussed in Section 2.1.4.

For 2-D planar (IAX=1) or axisymmetric (IAX=2) flow problems, the source terms of the momentum equations reduce to:
where \( \lambda_a \) is the coordinate rotation speed about the X-axis, Y-axis, and Z-axis, respectively. The direction of gravitational and buoyancy forces are designated to be in negative and positive Y-axis, respectively. In the FDNS-RFV/PVM code, the option to account for the buoyancy force in thermally driven gravity flows is currently available for incompressible flows only, while the option to include the gravitational force directly is available only in compressible flow calculations. For thermally driven flows, the buoyancy force, \( B_y \), is calculated as

\[
B_y = \frac{g \beta T_x X_\infty}{U_x^2} \rho T^* = \frac{\text{Gr}}{\text{Re}^2} \rho T^* \quad \text{where} \quad T^* = \frac{T - T_\infty}{T_\infty}
\]

where \( \beta \) is the coefficient of thermal expansion, \( g \) is the gravity, \( \text{Gr} \) is the Grashof number, \( \text{Re} \) is the Reynolds number, and \( U_4, X_4, \) and \( T_4 \) are the freestream (reference) velocity, length scale, and temperature, respectively.

**2.1.1 Homogeneous Fluid Model**

In order to analyze the effect of phase changes in the spray combustion flows, real-fluid thermal and caloric equations of state (EOS) were developed. Hirschfelder's EOS (HBMS)\(^1\) was selected to solve the real fluid properties because it treats the gas, dense gas and liquid regions separately and produces reasonable accuracy over a wide range of reduced pressures and temperatures. The thermodynamics equations have the form

\[
\frac{P}{P_{c}} = \sum_{j=1}^{4} T_i^{j} \rho^{-\frac{2}{i}} \sum_{i=1}^{6} B_{ij} \rho^{-\frac{2}{i}} \quad \text{where} \quad T_r = \frac{T}{T_c} \quad \text{and} \quad \rho_r = \frac{\rho}{\rho_c}
\]
where $P_c$, $T_c$, $\rho_c$, and $Z_c$ are the pressure, temperature, density, and compressibility at the critical condition. $B_{ij}$ are the coefficients of the thermal property polynomial for a given species for each of 3 single-phase regions. $H$ and $H_0$ are the real-fluid enthalpy and ideal-gas enthalpy for a given species, and $R$ is the gas constant. These equations are not only of sufficiently high order that properties are accurately predicted for a wide range of conditions, but component submodels may be easily modified. In this instance, the vapor pressure curve and the liquid phase density correlations were improved over the original HBMS formulation. Multicomponent mixtures properties were calculated by either adding partial specific volumes or partial pressures. The partial volume methodology is essential to provide accurate and robust predictions when a small amount of multi-component vapor and a large amount of liquid, or vice versa, are present at the same location. The properties routines also include correlations for the transport properties. A data base file of thermodynamics properties for many chemical species (dbase.dat) is constructed and included as a part of the FDNS-RFV/PVM source code. The data include fluid properties at the critical point and the normal boiling point, compressibility factor, and curve-fitting parameters for various species.

2.1.2 Pressure Correction Equation

Since the predictor-corrector methodology is utilized in the FDNS-RFV/PVM code, a correlation is needed to link the velocity changes and the pressure changes. The correlation is derived from an approximate, discretized momentum and continuity equations. A simplified perturbed momentum equation (velocity-correction equation) can be written as:

$$u_i' \approx D_p \nabla p'$$

where $D_p$ is proportional to the magnitude of the matrix coefficients of the momentum equations for a given grid point. Next, for deriving a pressure-correction equation, the velocity and density fields are perturbed and expanded in the continuity equation based on the equation of state. In order to use both the ideal gas and real fluid models to solve the continuity equation, the correlation between density and pressure changes used in the pressure-correction equation needs to be modified to account for the compressibility effect. For the traditional predictor-corrector method, the correlation between density and pressure changes was constructed based on the perfect gas law. This is no longer valid if the real fluid model is employed to simulate spray combustion flows. Hence, a new correlation is obtained by using a constant-temperature derivative, which is

$$\rho' = p' \left( \frac{d \rho}{d p} \right)_T = p' \gamma \left( \frac{d \rho}{d p} \right)_s = \beta_p p'$$

where for the perfect gas:

$$\beta_p = \frac{1}{R T}$$

and for the real-fluid model:

$$\beta_p = \frac{\gamma}{a^2}$$

where $\gamma$, $R$ and $a$ are the specific heat, gas constant and sonic speed of the flow mixture, respectively.
By neglecting higher order terms and utilizing the velocity-correction relation above, a pressure-correction equation suitable for both compressible and incompressible flows is obtained, and can be written as

\[
\frac{\beta_p}{\Delta t} p' + \nabla \cdot ( u_i \beta_p p') - \nabla \cdot ( \rho^* D_p \nabla p') = -\nabla \cdot ( \rho^* u_i ) - \frac{p^* - p^n}{\Delta t}
\]

The superscripts of * and n denote the value at the intermediate and previous time steps. A dissipation term based on density field is added to the right hand side of the above equation near shock waves to reduce oscillations in pressure. Once a solution of the above equation is obtained, the velocity field and the pressure field are updated through the velocity-correction relation and the following equation, respectively.

\[
p^{n+1} = p^n + p'
\]

The density field and temperature field are then updated by applying the equation of state. For the multi-corrector procedure, the above steps are repeated for about 4 to 6 times to make certain that a continuity-satisfied flow field is obtained at the end of every time step.

### 2.1.3 Wall Function Model

A modified wall function approach is employed to provide good near-wall approximation which is less sensitive to the near-wall grid spacing. In this approach, the profile of the non-dimensionalized velocity \( u^+ \) is formulated based on the velocity profile suggested by Liakopoulos\(^14\), which is expressed as

\[
u^+ = \ln \left[ \frac{( y^+ + 11 )^{4.02}}{( y^+^2 - 7.37 y^+ + 83.3 )^{0.79}} \right] + 5.63 \tan^{-1} \left( 0.12 y^+ - 0.441 \right) - 3.81
\]

where \( u^+/u_r \), \( y^+ = \rho u_r y/\mu \), and \( u_r = \sqrt{\tau_\text{w}/\rho} \). This velocity profile provides a smooth transition between logarithmic law of the wall and linear viscous sublayer variation. In addition, a modified wall function treatment\(^15\), which is suitable for compressible flows for accounting for the aerodynamic heating effects, was also incorporated in the FDNS-RFV/PVM code to provide appropriate heat transfer calculation near wall boundaries. The wall heat flux is formulated as

\[
q_w = \frac{\rho u_r}{Pr_t} u^+ \left[ h_w - h_p - \sqrt{Pr_t} \left( \frac{V_p - V_w}{2} \right)^2 \right]
\]

where \( V \) is the total velocity, \( Pr_t \) is the turbulent Prandtl number, and the subscripts p and w denote values at the wall function and wall points, respectively. For turbulent flows, the square root in this equation should be replaced with a cubic root. For adiabatic wall conditions, the wall temperature is calculated by assigning zero wall heat flux to the above equation.

### 2.1.4 Compressible Turbulence Models

Three compressibility corrected k-\( \varepsilon \) turbulence models, namely k-corrected\(^16\), \( \varepsilon \)-corrected\(^17\) and temperature correction\(^18\) models, are included. The basic function of the compressibility corrections is to suppress turbulence mixing (i.e. reduce eddy viscosity) for high Mach number and/or high temperature flows. These three correction models are:
(1) k-corrected Model

In the k-corrected model, an additional dissipation term is added to the k-equation to reduce the turbulence mixing at the hot and fast flow region. This additional term is based on the local turbulence Mach number. The compressibility corrected k-equation can be written as:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_j k}{\partial x_j} = \frac{\partial}{\partial x_j} \left( \frac{\mu + \mu_t}{\sigma_k} \frac{\partial k}{\partial x_j} \right) + \rho \left( P_t - \varepsilon \right) - \rho M_t^2 \varepsilon$$

where $M_t = \sqrt{\frac{k}{a}}$, represents the turbulence Mach number used to provide compressibility correction. However, this model is less effective than the following $\varepsilon$-corrected model.

(2) $\varepsilon$-corrected Model

In the $\varepsilon$-corrected model, the production rate modeling constant in the $\varepsilon$-equation is re-scaled by a function depending on the local flow Mach number. In this correction model, the turbulence dissipation rate is enhanced at the region of high flow Mach number, and thus the turbulence mixing is suppressed. This model can effectively account for the compressibility effect for the high speed and high temperature flows, such as nozzle flows and plumes. The corrected $\varepsilon$-equation is written as:

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho u_j \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j} \left( \frac{\mu + \mu_t}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial x_j} \right) + \rho \frac{\varepsilon}{k} \left[ C_1 \left( 1 + 0.08 M^{0.25} \right) P_t + C_3 \frac{P_t^2}{\varepsilon} - C_2 \varepsilon \right]$$

as:

where $M$ is the local flow mach number.

(3) Temperature correction Model

In the temperature correction model, another modeling constant of the production term in the $\varepsilon$-equation is modified to be a function depending on the local flow temperature. This correction model is particularly effective for the low speed, high temperature flowfield such as flows in the combustion chamber. In this model, the production term is modified such that the turbulence dissipation rate, $\varepsilon$, is increased as the local flow temperature is raised, and can be expressed as

$$C_3^* = \frac{\rho P_t^2}{k}; \text{ and } C_3^* = C_3 \left( \frac{T}{T_a} \right)^{\lambda}$$

where $T$ and $T_a$ are the local flow temperature and ambient temperature, respectively. The exponent, $\lambda$, varies for different reaction models (typically 0.4 for wet-CO mechanism, and 0.6 for H$_2$/O$_2$ reaction).

2.1.5 Chemistry Models

In the FDNS-RFV/PVM code, both finite-rate chemistry and equilibrium chemistry are incorporated in the code to compute reacting flow problems. A general system of chemical reactions
can be written in terms of its stoichiometric coefficients \((\nu_{ij}'\) and \(\nu_{ij}''\)) and the \(i\)-th chemical species \((M_i)\) of the \(j\)-th reaction as

\[
\sum_i \nu_{ij}' M_i \leftrightarrow \sum_i \nu_{ij}'' M_i''
\]

1) **Finite-Rate Chemistry:**

The transport equation of the \(i\)-th species mass fraction \((\alpha_i)\) can be written as

\[
\rho \frac{D\alpha_i}{Dt} = \nabla \cdot \left( \frac{\mu + \mu_t}{\sigma_\alpha} \nabla \alpha_i \right) = \omega_i
\]

where \(\sigma_\alpha\) represents the Schmidt number for turbulent diffusion, and \(\omega_i\) is the production rate of species \(i\), which can be expressed as

\[
\omega_i = M_{W_i} \sum_j (\nu_{ij}'\nu_{ij}'' - \nu_{ij}'\nu_{ij}'') (R_{f_j} - R_{b_j})
\]

where

\[
R_{f_j} = K_{f_j} \prod_i \left( \frac{\rho \alpha_i}{M_{W_i}} \right)^{\gamma_{ij}'}; \quad R_{b_j} = K_{b_j} \prod_i \left( \frac{\rho \alpha_i}{M_{W_i}} \right)^{\gamma_{ij}''}
\]

In the above equation, \(M_{W_i}\) is the molecular weight of species \(i\), \(K_{f_j}\) and \(K_{b_j}\) are the forward and backward rates of reaction \(j\), and \(\gamma_{ij}'\) and \(\gamma_{ij}''\) are the power dependencies of reactants and products, respectively. The backward reaction rate can be calculated from the forward reaction rate and a chemical equilibrium constant, \(K_{eq}\), i.e. \(K_{b_j} = K_{f_j}/K_{eq}\).

Due to the stiffness of the finite-rate equation, several methods were used to solve the equations. At first, an explicit method with a penalty function was used, and the system of species continuity equations were then solved by employing a small time step size based on the assigned tolerance for species mass fraction correction within a time step, which is

\[
(\Delta t)_{chemistry} = \rho (\Delta \alpha_i)_{assigned} / \omega_i
\]

This approach is very efficient in a pre-mixed flow environment\(^5,7\), where elemental species are fairly constant throughout the flowfield. However, this method fails to accurately describe non-premixed flows because the elements may not be conserved. Various methods (first-/second-order implicit or second-/fourth-order parasol) were therefore employed to linearize the species production rate equations, and thus evaluate the species production rate. The species production rate equation can be expressed in terms of the molar density \(C_i (= \rho \alpha_i/M_{W_i})\)

\[
\frac{\omega_i}{M_{W_i}} = \frac{\Delta C_i}{\Delta t} = \sum_j (\nu_{ij}'' - \nu_{ij}') (R_{f_j} - R_{b_j}) = S_i
\]

The linearization process was demonstrated in Reference 3. The formulation for calculating the
change of species concentrations with various orders of accuracy can be summarized in a general form

\[ \omega_i = \Delta C_i^{n+1} \frac{M_w}{\Delta t} \]

where \( \phi_1, \phi_2 \) and \( \phi_3 \) for the various schemes are summarized in Table 2.3. By solving the above equation, the change of species concentration can be obtained for a given time interval. However, the time step for the flow solver can be too large compared to the reaction time of the finite rate chemistry and lead to over- or under-predicted species changes. Thus, for a given flow time step, the chemistry time steps will be divided into several sub-intervals. The overall species changes will be the integration of the species changes over each sub-intervals. Once the overall species changes are obtained, the average species production rates can be calculated based on the species changes and their corresponding flow time step. In this approach, the average species production rates depend on the selected flow time step size. The flow time step should not be too large; otherwise, in complex reacting flows temperature and concentration gradients may become so large that they are affected by the time step size. A numerical experiment was conducted to demonstrate the sensitivity to the flow time step size. Though the computation time is increased substantially compared to the explicit formulation, the species elements and concentrations are conserved, no matter what types of flowfield or level of stiffness in the combustion kinetics models is encountered.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>( \phi_1 )</th>
<th>( \phi_2 )</th>
<th>( \phi_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st-order implicit</td>
<td>( \Delta t )</td>
<td>( \Delta t )</td>
<td>0</td>
</tr>
<tr>
<td>2nd-order implicit</td>
<td>( \Delta t )</td>
<td>( \Delta t/2 )</td>
<td>0</td>
</tr>
<tr>
<td>2nd-order Parasol</td>
<td>( \Delta t )</td>
<td>( \Delta t/2 )</td>
<td>0</td>
</tr>
<tr>
<td>4th-order Parasol</td>
<td>( \Delta t )</td>
<td>( \Delta t/2 )</td>
<td>( -\Delta t^2/12 )</td>
</tr>
</tbody>
</table>

(2) Wet CO Equilibrium Chemistry:

The wet-CO mechanism, consisting of 8 species, may be modeled using 3 element balances:

\[
\begin{align*}
E_H &= 2N_{H_2O} + 2N_{H_2} + N_H + N_{OH} \\
E_O &= N_{H_2O} + 2N_{O_2} + N_O + N_{CO} + 2N_{CO_2} + N_{OH} \\
E_C &= N_{CO} + N_{CO_2}
\end{align*}
\]

and the following 5 reactions:
where $K_j$ is the equilibrium constant for reaction "j", and $N_i = \rho \alpha_i / M_w_i$ is the molar density of species "i", where $\alpha_i$ and $M_w_i$ are the mass fraction and molecular weight of species "i", respectively. The reaction set is not unique, but all other sets can be derived from this set. The above equations were solved to obtain equilibrium species concentrations.

### 2.1.6 Particulate Multi-phase Model

The present method solves the particle equations of motion and the particle energy equation on the Lagrangian framework. Each particle group trajectory is tracked by integrating the particle equations of motion. The inter-phase drag forces and heat transfer fluxes are stored and included in the gas phase governing equations. The methodology in the current version is for steady-state applications only. Transient particulate models will be released in future publications. The particle momentum and energy equations are written as: (Al$_2$O$_3$ particle properties are used in the present model.)

$$\frac{D V_i}{D t} = \frac{(U_i - V_i)}{t_d}$$

$$\frac{D h_p}{D t} = C_{p_v} \left( \frac{T_{aw} - T_p}{t_h} \right) - \frac{6 \sigma \varepsilon f T^4}{(\rho d)_p}$$

where
U_i = Gas Velocity
V_i = Particle Velocity

t_d = Particle Dynamic Relaxation Time = \frac{3 \rho_p d_p}{3 C_d \rho_g | U_i - V_i |}

h_p = Particle Enthalpy
T_o = Particle Temperature
T_{aw} = Gas Recovery Temperature

T_H = Particle Thermal- Equilibrium Time = \frac{\rho_p d_p P_{r}}{12 \text{ Nu} \mu}

C_{pt} = Gas Specific Heat
\rho_g = Gas Density
\sigma = Stefan- Boltzman Constant
\varepsilon = Particle- Emissivity = 0.2 to 0.3
f = Radiation Interchange Factor
2.2 NUMERICAL SCHEMES

In the present CFD methodology, finite difference approximations are employed to discretize the transport equations on non-staggered grid mesh systems. High-order upwind, TVD, or central difference schemes plus adaptive second-order and fourth-order dissipation terms are used to model the convective terms. Second-order central difference schemes are used for the viscous and source terms of the governing equations. A first-order upwind difference scheme is employed for the convection process of positive-definite scalar quantities (e.g. turbulence quantities and species mass fractions, etc.). A pressure based predictor/multi-corrector solution procedure is employed in the FDNS-RFV/PVM code to enhance velocity-pressure coupling and continuity-satisfied flowfield solutions at the end of every time step. Successful results of viscous flow computations using the present pressure based method have been reported\textsuperscript{3-10}. For incompressible flows, an elliptic pressure correction equation is used for divergence-free velocity-pressure coupling. As the flow speed is increased, a hyperbolic pressure correction equation results due to the perturbation in the density field. This allows the flow solution to follow the characteristics of high speed flows. A second-order accurate time-centered time-marching scheme is used for the temporal discretization for transient flow simulations. For steady state flow problems, an implicit Euler time-marching scheme can be used for efficient turn-around. The selection of the upwind and time-marching schemes is done in the input data file.

2.2.1 Adaptive Dissipation Scheme

For simplicity, let us consider fluxes in the $\xi$-direction (or i-direction) only. The convection flux can be discretized as:

$$\frac{\partial \rho U_i q_i}{\partial \xi} = \frac{\partial F}{\partial \xi} \approx (F_{i+1/2} - F_{i-1/2}) - (d_{i+1/2} - d_{i-1/2})$$

The first term on the right hand side of the above equation represents the baseline first-order upwind difference scheme, which is

$$F_{i+1/2} = \max \{ 0, (\rho U)_{i+1/2} \} q_i + \max \{ 0, -(\rho U)_{i+1/2} \} q_{i+1}$$
$$F_{i-1/2} = \max \{ 0, (\rho U)_{i-1/2} \} q_{i-1} + \max \{ 0, -(\rho U)_{i-1/2} \} q_i$$

and the second term on the right hand side of the discretized equation is the adaptive dissipation term, which includes the fourth-order dissipation term (D), and the anti-damping term (A) which is employed to obtain high-order upwind, TVD, or central difference schemes. The fourth-order dissipation term is only used for second-order and third-order upwind, and central difference schemes. The adaptive dissipation term can be expressed as

$$d_{i+1/2} = |\rho U|_{i+1/2} \left[ (1 - \text{REC}) A_{i+1/2} + 0.05 \alpha_d D_{i+1/2} \right]$$
$$d_{i-1/2} = |\rho U|_{i-1/2} \left[ (1 - \text{REC}) A_{i-1/2} + 0.05 \alpha_d D_{i-1/2} \right]$$

The presence of the fourth-order dissipation term is controlled by the shock monitoring parameter, $\alpha_d$, which is defined as:

$$\alpha_d = \max \{ 0, 1 - 25 \max (\psi_{i-1}, \psi_i, \psi_{i+1}) \}$$
$$\psi_i = |p_{i-1} - 2p_i + p_{i+1}| / (p_{i-1} + 2p_i + p_{i+1})$$

It is obvious that $\alpha_d$ will have a value of zero when a very large pressure gradient occurs, and thus the baseline first-order upwind difference or high-order TVD schemes would be utilized. The forth-
order dissipation term can be expressed as

\[ D_{i+1/2} = - \left[ q_{i+1/2} \right]_{i+1/2} = 2 \Delta q_{i+1/2} - \Delta q_{i+3/2} - \Delta q_{i-1/2} \]
\[ D_{i-1/2} = - \left[ q_{i-1/2} \right]_{i-1/2} = 2 \Delta q_{i-1/2} - \Delta q_{i+1/2} - \Delta q_{i-3/2} \]

and \( \Delta q_{i+1/2} = q_{i+1} - q_i \) and etc. The upwind parameter, REC, a user-specified parameter in the input file is used to control the amount of first-order upwind scheme used for the convection term. Values of REC close to 1, corresponding to large dampening, are used to start calculations. This value should be reduced to approximately 0.1 as the solution progresses to obtain higher order of accuracy.

The anti-damping term \( A \) is in a different form such that it can be combined with the baseline first-order upwind difference scheme to obtain high-order upwind, TVD, or second-order central difference schemes. In the present approach, the anti-damping terms are formulated as follows.

(1) Second-order Upwind Scheme:

\[ A_{i+1/2} = -0.5 \alpha_d \begin{cases} 2/3 \Delta q_{i+1/2} + 1/3 \Delta q_{i-1/2}, & \text{if } U_{i+1/2} > 0 \\ 2/3 \Delta q_{i+1/2} + 1/3 \Delta q_{i+3/2}, & \text{if } U_{i+1/2} < 0 \end{cases} \]
\[ A_{i-1/2} = 0.5 \alpha_d \begin{cases} 2/3 \Delta q_{i-1/2} + 1/3 \Delta q_{i-3/2}, & \text{if } U_{i-1/2} > 0 \\ 2/3 \Delta q_{i-1/2} + 1/3 \Delta q_{i+1/2}, & \text{if } U_{i-1/2} < 0 \end{cases} \]

(2) Third-order Upwind Scheme:

\[ A_{i+1/2} = 0.5 \alpha_d \begin{cases} 2/3 \Delta q_{i+1/2} + 1/3 \Delta q_{i-1/2}, & \text{if } U_{i+1/2} > 0 \\ 2/3 \Delta q_{i+1/2} + 1/3 \Delta q_{i+3/2}, & \text{if } U_{i+1/2} < 0 \end{cases} \]
\[ A_{i-1/2} = 0.5 \alpha_d \begin{cases} 2/3 \Delta q_{i-1/2} + 1/3 \Delta q_{i-3/2}, & \text{if } U_{i-1/2} > 0 \\ 2/3 \Delta q_{i-1/2} + 1/3 \Delta q_{i+1/2}, & \text{if } U_{i-1/2} < 0 \end{cases} \]

(3) Second-order Central Scheme:

\[ A_{i+1/2} = -0.5 \alpha_d \Delta q_{i+1/2} \]
\[ A_{i-1/2} = -0.5 \alpha_d \Delta q_{i-1/2} \]

(4) Upwind TVD Scheme:

\[ A_{i+1/2} = \frac{1}{4} \begin{cases} \Psi_{i+1/2}^+ + \Psi_{i-1/2}^+ + \alpha_t (\Psi_{i+1/2}^+ - \Psi_{i-1/2}^-), & \text{if } U_{i+1/2} > 0 \\ \Psi_{i+1/2}^- + \Psi_{i+3/2}^+ + \alpha_t (\Psi_{i+1/2}^- - \Psi_{i+3/2}^-), & \text{if } U_{i+1/2} < 0 \end{cases} \]
\[ A_{i-1/2} = \frac{1}{4} \begin{cases} \Psi_{i-1/2}^+ + \Psi_{i-3/2}^+ + \alpha_t (\Psi_{i-1/2}^+ - \Psi_{i-3/2}^-), & \text{if } U_{i-1/2} > 0 \\ \Psi_{i-1/2}^- + \Psi_{i+1/2}^+ + \alpha_t (\Psi_{i-1/2}^- - \Psi_{i+1/2}^-), & \text{if } U_{i-1/2} < 0 \end{cases} \]
where the minmod functions in the TVD flux limiters are written as

$$\Psi_{i+1/2}^{\pm} = \text{sign} \left( \Delta q_{i+1/2} \right) \max\{0, \min \left[ |\Delta q_{i+1/2}|, \alpha_c \text{sign} \left( \Delta q_{i+1/2} \right) \Delta q_{i+1/2} \right]\}$$

and $\alpha_c = \frac{3 - \alpha_t}{1 - \alpha_t}$; where $\alpha_t = \begin{cases} -1, & \text{for 2nd-order upwind} \\ 1/3, & \text{for 3rd-order upwind} \end{cases}$

The selection of the above high-order scheme depends on the specification of an input parameter, IREC. Table 2.4 summarizes the upwind options.

<table>
<thead>
<tr>
<th>2nd-order upwind</th>
<th>3rd-order upwind</th>
<th>2nd-order Central</th>
<th>2nd-order upwind TVD</th>
<th>3rd-order upwind TVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>IREC</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

### 2.2.2 Viscous Term Discretization

All viscous fluxes are discretized using the second-order central differencing schemes. The diffusion terms are divided into orthogonal and non-orthogonal parts. The orthogonal terms are treated implicitly and the non-orthogonal terms are lumped into the explicit part of the source terms. This arrangement reduces the computer memory requirements compared to the fully implicit treatment. A fully implicit arrangement can be made available by introducing more matrix element arrays into the FDNS-RFV/PVM code. For simplicity, let us consider the viscous fluxes in the i-direction.

$$\frac{\partial}{\partial \xi} \left( \mu_e G_{ij} \frac{\partial q}{\partial \xi} \right)_{i+1/2} = \left( \mu_e G_{i11} \frac{\partial q}{\partial \xi} \right)_{i+1/2} - \mu_e G_{i11} \frac{\partial q}{\partial \xi} \mu_e G_{i12} \frac{\partial q}{\partial \eta} \mu_e G_{i13} \frac{\partial q}{\partial \zeta}$$

where the viscous terms associated with $G_{i11}$ are orthogonal terms and the viscous terms involving $G_{i12}$ and $G_{i13}$ represent non-orthogonal terms. These viscous fluxes on the $i+1/2$ control volume interface can be written as:

1. Orthogonal terms = $\left( \mu_e G_{i11} \right)_{i+1/2} \left( q_{i+1} - q_i \right)$

2. Non-orthogonal terms = $0.25 \left( \mu_e G_{i12} \right)_{i+1/2} \left( q_{i+1, j+1, k} + q_{i, j+1, k} - q_{i+1, j-1, k} - q_{i, j-1, k} \right) + 0.25 \left( \mu_e G_{i13} \right)_{i+1/2} \left( q_{i+1, j, k+1} + q_{i, j, k+1} - q_{i+1, j, k-1} - q_{i, j, k-1} \right)$

For 2-D flow problems, viscous terms involving $G_{i13}$ vanish. As one can see, a fully implicit viscous formulation would greatly increase the matrix size (i.e. 4 more coefficients for 2-D cases and 12 more coefficients for 3-D cases) for every equation.
2.2.3 Time-Marching Scheme

For time accuracy, an efficient non-iterative time-centered time-marching scheme with a multi-corrector solution algorithm is employed. First of all, the governing equations are linearized by applying the aforementioned finite difference discretization schemes to the flux and source terms. A system of linearized algebraic equations are obtained as a result of the linearization. A relaxation solution procedure (i.e. the linearized algebraic equations are solved sequentially with an iterative full matrix solver) is employed for the solution of the governing equations. For convenience, the conservation equation can be written as:

\[
\frac{1}{J} \frac{\partial \rho q}{\partial t} = \frac{\partial F_i}{\partial \xi_i} + S_q = R_q
\]

or, in finite difference form,

\[
\frac{1}{J \Delta t} \left[ (\rho q)^{n+1} - (\rho q)^n \right] = \theta R_q^{n+1} + (1 - \theta) R_q^n
\]

where the superscripts \( n \) and \( n+1 \) denote the current and the next time levels respectively. \( \theta \) is a time-marching control parameter, which is specified in the input data file. \( \theta = 1 \) and \( \theta = 1/2 \) are for an implicit Euler time-marching and a time-centered time-marching schemes, respectively. The following linearization is then incorporated.

\[
(\rho q)^{n+1} = (\rho q)^n + \rho^n \Delta q^n
\]

\[
R_q^{n+1} = R_q^n + \left( \frac{\partial R_q}{\partial q} \right)^n \Delta q^n
\]

With these relations, a delta form of the time-marching equation can be written as:

\[
\left( \frac{\rho}{J \Delta t} \theta \frac{\partial R_q}{\partial q} \right)^n \Delta q^n = R_q^n
\]

This system of equations will then be solved by using the matrix solver specified by the user.

2.3 BOUNDARY CONDITIONS

The FDNS-RFV/PVM code provides options (mostly through input data specifications) for treating various types of boundaries (e.g. inlet, outlet, symmetry, periodic, freestream, singularity lines and solid wall boundaries with and/or without blowing) and the location of each boundary. User defined boundary conditions to override the existing ones can be provided in one of the include files (i.e. fmain02) with proper FORTRAN programming. The input data controlled boundary conditions are described below.

2.3.1 Inlet Flow Boundaries

For incompressible inlet boundaries, only the pressure waves are extrapolated upstream. For
subsonic inlet boundaries, two types of inlet boundary conditions can be specified. They are: (1) fixed inlet total conditions (i.e. type 1 in the input data); and (2) fixed mass flow rate inlet condition (i.e. type -1 in the input data). The second subsonic inlet boundary condition (type -1) is usually used for mass injected inlet boundaries such as near the propellant burning surface of a solid rocket motor combustion chamber. For supersonic inlet boundaries, all the flow variables are fixed at specified values.

2.3.2 Exit Flow Boundaries

For outlet boundaries, all variables are extrapolated downstream in the first step. A special adaptive gradient detection extrapolation method is employed for the velocity vectors and pressure and temperature fields to provide smooth wave propagation through the outlet boundaries. For subsonic or incompressible outlet boundaries, two options are provided: (1) outlet velocity vectors are corrected based on the global mass conservation conditions (i.e. PRAT = 0.0 in the input data); and (2) outlet pressure profile is updated such that the ratio of the pressure of a outlet pressure reference point (IPEX, JPEX in the input data) to the atmospheric pressure (14.7 psi) is kept at a specified value (i.e. PRAT = pressure ratio). IPEX represents the global grid number of the selected grid point in zones JPEX. The zonal and global grid numbering system used throughout the FDNS-RFV/PVM code are calculated using the following formulas respectively.

Zonal: \[ III = I + (J - 1) \times JZS(NZ) + (K - 1) \times KZS(NZ), \] for zone NZ

Global: \[ IJK = IZS(NZ) + III \]

where IZS(NZ), JZS(NZ) and KZS(NZ) are the grid index incremental counts for zone number NZ, which are calculated based on the input grid sizes, \( I_{\text{max}} = IZT(NZ), J_{\text{max}} = JZT(NZ), \) and \( K_{\text{max}} = KZT(NZ). \)

The first method is mainly used for incompressible or subsonic internal flow problems. The second method can be used for external and internal subsonic outlet boundaries. It is important that for incompressible flow cases, the global pressure reference point (IPC, JPC) must be an interior point (i.e. not a boundary point), where IPC represents the global grid number of the selected grid point in zones JPC. For supersonic outlet boundaries, only extrapolated conditions are employed (i.e. PRAT = -1.0). For scalar variables such as turbulence quantities and species concentrations, the two kinds of outlet extrapolation methods are available. They are: zero gradient extrapolation (i.e. \( IEXX = 1 \)); and linear extrapolation (i.e. \( IEXX = 2 \)). The zero gradient scalar extrapolation method is recommended for reacting flow applications to provide better solution stability.

2.3.3 Symmetry Boundaries

For symmetry boundary conditions on symmetry planes, symmetry lines of 2-D cases or inviscid slip boundaries, zero normal gradients for all scalar quantities are specified. Special treatment for the velocity vectors are provided to reflect zero mass flux condition on the symmetry planes. The velocity vectors are first projected onto the symmetry planes. The surface normal components of the vectors are then assigned to be zero such that the resultant vectors are tangent to the symmetry planes.

2.3.4 Zonal Interface Boundaries
For multi-zone interface boundary conditions, two methods are available. In the first method, grid lines must be continuous across patched zonal interface. The zonal patching index specification is given in through the input data file. The second zonal method allows the use of non-continuous patched or non-overlaid mesh systems. In this case, the user needs to implement the subroutine INFACE (in fl.f module) to conduct the zonal interpolation index identification, grid movement and zonal interpolation. The subroutine INFACE is currently empty in the FDNS-RFV/PVM code. For turbo-machinery applications using periodic boundaries, special input data specification must be used (i.e. ICYC = 3 and IGEO = 9).

2.3.5 Singularity Boundaries

An averaging procedure along singularity lines is provided in the FDNS-RFV/PVM code to circumvent possible numerical difficulties for resolving the flow field near singularity lines. The flow solution on the singularity lines is assumed to take the averaged values of the surrounding points. Additional conditions applied to the singularity lines can then be treated explicitly in the include file fmain02.

Additional boundary conditions such as freestream inlet flow angle extrapolation, jet outlet pressure condition updating and time dependent inlet and/or wall boundary conditions can be implemented by adding program coding in one of the include files (i.e. fmain02) in the main program. For program restart flow condition modifications, another main program include file (i.e. fmain01) can be utilized.

2.3.6 Wall Boundaries

The FDNS-RFV/PVM code provides a multiple solid-wall blocking feature which allows the user to specify wall elements anywhere inside the flow domain. The wall surface orientations (direction cosine) are also calculated and used for wall function modeling and wall extrapolation purpose. For viscous flow computations, non-slip boundary conditions are employed for the momentum equations. A standard wall function approach with a modified universal velocity profile is employed for turbulent flow computations. Fixed wall temperature distributions or adiabatic wall boundary conditions are the two wall boundary conditions available for the energy equation (i.e. IWTM = -1 and 1 respectively). Since the current version provides the option of running conjugate heat transfer between solid-wall and fluid, fixed wall temperature condition (IWTM = -1) is set by the code when conjugate heat transfer option is activated (i.e. IWALL = 1 to activate and IWALL = 0 to deactivate). Pressure along the wall is evaluated by using extrapolation. Since the FDNS-RFV/PVM code is designed for generalized coordinate systems, non-orthogonal boundary grid effects are also taken into account when normal gradients at the solid wall surface are evaluated. Let us consider a wall surface on the i-k plane and at j = 1 with local direction cosines of the normal vector defined as: cos \( \alpha \), cos \( \beta \), and cos \( \gamma \). The local zero normal gradient condition for the flow variable \( q \) can be written as:

\[
\frac{\partial q}{\partial n} = 0 = q_x \cos \alpha + q_y \cos \beta + q_z \cos \gamma
\]

where
The wall boundary values are calculated using the above equation. That is, the quantities, $q_\xi$ and $q_\zeta$, are evaluated on $j = 1$ plane. This is a good approximation providing that the grid variation away from the wall surfaces is smooth. The above zero normal gradient treatment is performed explicitly at the end of each time step.

### 2.4 INITIAL CONDITIONS

The mesh systems and flow field initial guesses for running the FDNS-RFV/PVM code can be prepared in two ways. The first method is used in setting up some sample cases later in this manual. Using this method, grid and initial flow generation FORTRAN codes are written to generate grid and initial flow field data files (file unit fort.10 in the sample cases). These data files are then read in from the example include file, fexmp01, when the calculation is started using the example start option (IDATA = 2). The second method involves the preparation of a grid and flow field restart files (fort.12 and fort.13 respectively). Then, the FDNS-RFV/PVM code is started using the restart option (IDATA = 1 or 0). To use the second method, the following restart file data format (Tables 2.5 and 2.6) must be used in preparing the data files of grid meshes and the initial flowfield.

The FDNS-RFV/PVM code uses grid systems that follow the right hand rule for the $I$, $J$- and $K$-line orientations such that the cell volumes are always positive. The grid cell volumes are calculated and checked in subroutine TRANF. On detection of any zero or negative volume, a warning message is issued and the program stops. To avoid this message for singularity lines where surfaces collapse into lines, small finite radii (1E-06 for instance) must be used for generating the singularity line surfaces (e.g. centerlines of pi segments). It is also recommended that the grid used should have smooth Jacobian variations so that grid effects can be minimized.
### Table 2.5 Data Format of the Restart Grid File

<table>
<thead>
<tr>
<th>WRITE(12,1)</th>
<th>IZON</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO IZ=1,IZON</td>
<td></td>
</tr>
<tr>
<td>WRITE(12,1)</td>
<td>IZT(IZ),JZT(IZ),KZT(IZ)</td>
</tr>
<tr>
<td>ENDDO</td>
<td></td>
</tr>
<tr>
<td>DO IZ=1,IZON</td>
<td></td>
</tr>
<tr>
<td>I2=IZT(IZ)</td>
<td></td>
</tr>
<tr>
<td>J2=JZT(IZ)</td>
<td></td>
</tr>
<tr>
<td>K2=KZT(IZ)</td>
<td></td>
</tr>
<tr>
<td>WRITE(12,2)</td>
<td>(((X(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>WRITE(12,2)</td>
<td>(((Y(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>WRITE(12,2)</td>
<td>(((Z(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>ENDDO</td>
<td></td>
</tr>
<tr>
<td>1 FORMAT(15I5)</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
1. IZON: the total number of grid zones.
2. IZT(IZ), JZT(IZ), and KZT(IZ): maximum grid numbers in the I-, J-, and K-direction for zone #IZ.
   (For 2-D flows, KZT(IZ)=1)
3. For 2-D axisymmetric flows, Z(I,J,K,IZ)=1.0; while for 2-D planar flows, Z(I,J,K,IZ) equals to 1.0 or the depth of the flow domain.
4. If IDATA = 0 (unformatted file option), then the format in each write statement should be eliminated.

### Table 2.6 Data Format of the Restart Flow File

<table>
<thead>
<tr>
<th>WRITE(13,3)</th>
<th>INSO(1),INSO(4),INSO(5),INSO(7),NGAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO IZ=1,IZON</td>
<td></td>
</tr>
<tr>
<td>I2=IZT(IZ)</td>
<td></td>
</tr>
<tr>
<td>J2=JZT(IZ)</td>
<td></td>
</tr>
<tr>
<td>K2=KZT(IZ)</td>
<td></td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((DEN(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((U(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((V(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((W(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((P(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>IF(INSO(4) .EQ. 1) THEN</td>
<td></td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((TM(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>IF(INSO(5) .EQ. 1) THEN</td>
<td></td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((DK(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((DE(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>IF(AMC .GT. 0.) THEN</td>
<td></td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((AM(I,J,K,IZ),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>IF(NGAS .GT. 0) THEN</td>
<td></td>
</tr>
<tr>
<td>DO KK=1,NGAS</td>
<td></td>
</tr>
<tr>
<td>WRITE(13,2)</td>
<td>(((FM(I,J,K,IZ,KK),I=1,I2),J=1,J2),K=1,K2)</td>
</tr>
<tr>
<td>ENDDO</td>
<td></td>
</tr>
<tr>
<td>END IF</td>
<td></td>
</tr>
<tr>
<td>2 FORMAT(5(1P,E16.8))</td>
<td></td>
</tr>
<tr>
<td>3 FORMAT(8I5)</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
Definitions of NGAS and INSO are detailed in Chapter 3.
DEN, U, V, W, P, TM, DK, DE, AM, Q, and FM are the flow density, flow velocities in X-, Y- and Z-axis, static pressure, static temperature, turbulent kinetic energy and its dissipation rate, local flow Mach numbers, fluid qualities, and species mass fractions, respectively. The flow Mach number and quality can have the value of unity as the initial guess. For 2-D axisymmetric flows, W (azimuthal velocity) can have non-zero value if the swirling component exists.
2.4.1 Reference Conditions

The initial flow variables should be non-dimensionalized by the reference conditions specified (depends on the parameter IUNIT which can be either 1 or 2) in the input data file (fort.11). If the reference quantities are chosen to be unity, then the system of the governing equations would be solved in dimensional form. The user must make sure that the non-dimensionalization process must be consistent. Either SI or English units can be used based on the user's preference. The reference conditions are defined below:

SI Unit: (for IUNIT = 1)
- Density : kg/m$^3$
- Velocity : m/sec
- Temperature : °K
- Length : m

English Unit: (for IUNIT = 2)
- Density : slugs/ft$^3$
- Velocity : ft/sec
- Temperature : °R
- Length : ft

If the compressibility option, ICOMP, in the input data file (fort.11) is specified to be greater than zero (or unity), then the compressible flow calculation will be conducted and the flow properties input to the code need to be normalized by the thermal properties of air at the reference conditions listed as follows, as well as specified in the input data file (fort.11).

SI Unit: (for IUNIT = 1)
- Reference Temperature (T$_{ref}$) : TREF = 300 °K
- Reference Gas Constant (R$_{ref}$) : RMXBAR = 288.5939026 J/kg-°K
- Reference Specific Heat (C$_{p,ref}$) : CPBAR = 1012.790527 J/kg-°K
- Reference Viscosity (µ$_{ref}$) : VISC = user specified (in N-s/m$^2$)
- Reference Pressure (P$_{ref}$) : PREF = user specified (in Pa)
- Reference Density ($\rho_{ref}$) : DENREF = calculated (in kg/m$^3$)
- Reference Velocity (U$_{ref}$) : UREF = user specified (in m/s)

English Unit: (for IUNIT = 2)
- Reference Temperature (T$_{ref}$) : TREF1 = 540 °R
- Reference Gas Constant (R$_{ref}$) : RMXBAR = 53.62715 ft-lb f/lbm-°R
- Reference Specific Heat (C$_{p,ref}$) : CPBAR = 188.199 ft-lb f/lbm-°R
- Reference Viscosity (µ$_{ref}$) : VISC = user specified (in lb f-s/ft$^2$)
- Reference Pressure (P$_{ref}$) : PREF = user specified (in psi)
- Reference Density ($\rho_{ref}$) : DNREF1 = calculated (in slugs/ft$^3$)
- Reference Velocity (U$_{ref}$) : UREF = user specified (in ft/s)

The correct value of the reference variables will be calculated in the code based on the following correlation.

Reference Density ($\rho_{ref}$) = $P_{ref}/(R_{ref} T_{ref})$
Reference Mach number \((M_{ref}) = AMC = \frac{U_{ref}}{a_{ref}}\)

\(a_{ref} = \text{reference speed of sound} = \left[\gamma_{ref} R_{ref} T_{ref}\right]^{0.5}\); and \(R_{ref} = \frac{R_{u}}{(M_w)_{ref}}\)

where \(\gamma_{ref} = 1.3985\), and \(R_{u}\) and \((M_w)_{ref}\) are the universal gas constant and the molecular weight of air, respectively. However, if the reference Mach number (AMC) in the input data file is set to 0, then the incompressible flow calculation will be activated and all the reference properties will be set to unity. In addition, the reference viscosity \((\mu_{ref})\) will be used to calculate local fluid viscosity base on Sutherland's Law:

\[
\frac{\mu}{\mu_{ref}} = \left(\frac{T}{T_{ref}}\right)^{0.7}
\]

### 2.4.2 Normalization of Flow Variables

The flow variables and grid coordinates are non-dimensionalized as:

- **Density:** \(\frac{\rho}{\rho_{ref}}\)
- **Velocity:** \(\frac{V}{U_{ref}}\)
- **Pressure:** \(\frac{P}{(\rho_{ref} U_{ref}^2)}\)
- **Temperature:** \(\frac{T}{T_{ref}}\)
- **Length:** \(\frac{L}{X_{ref}}\)
- **Viscosity:** \(\frac{\mu}{(\rho_{ref} U_{ref} X_{ref})}\)
- **Turbulence Kinetic Energy (TKE):** \(\frac{TKE}{U_{ref}^2}\)
- **Turbulence Dissipation Rate \((\varepsilon)\):** \(\frac{\varepsilon}{(U_{ref}^3 / X_{ref})}\)

Consistent units (either SI or English unit) must be used throughout the non-dimensionalization process. For turbulent flow applications, the turbulence kinetic energy \((k)\) and its dissipation rate \((\varepsilon)\) can be initialized using the nominal non-dimensional values when no measured data are available. That is,

\[
k = \frac{4}{3} \times (\text{Turbulence Intensity})^2
\]
\[
\varepsilon = 0.09 \times k^{1.5} / (0.03 \times \text{Characteristics Length})
\]

where the characteristics length can be a channel width or twice a boundary layer thickness, etc.

Besides the above reference quantities, there are two additional reference values, \(HH_{REF}\) and \(HH_{REF1}\), which are important for calculating wall heat fluxes. The wall heat fluxes are calculated using a non-dimensional heat flux variable, \(HTWN(I)\), stored at near-wall points (or wall function points). That is,

Wall Heat Flux = -HTWN(IJLO(J))*HHREF ---- for SI unit, Watts/cm²

or

Wall Heat Flux = -HTWN(IJLO(J))*HHREF1 ---- for English unit, Btu/sec-ft²

where IJLO(J) is a wall function point indices book-keeping array.
2.5 REFERENCES


3. FDNS-RFV/PVM CODE FEATURES

The FDNS-RFV/PVM code is a fully transparent and user friendly computational fluid dynamics code which is used to analyze a wide variety of fluid dynamics related engineering problems (e.g. internal and/or external flows with complex geometries, cases with laminar or turbulent flow conditions, and flows with ideal, real or reacting gas effects for all speed range -- incompressible to hypersonic flow regimes). For programming simplicity and computational efficiency, all the flow variables, except those in the subroutines for calculating chemical reaction source terms, are stored using COMMON blocks. There are 18 COMMON block include files: fdns01; fdns02; fdns03; ..........; fdns17; and fluid.inc. One-dimensional arrays are used for all flow variables representing two-dimensional or three-dimensional flow problems using structured single or multiple-block grids. The conversion between the (i, j, k) indices and the global 1-D indices is described in Section 2.3.

Before compiling the code, one must make sure that proper array dimensions are set in the first COMMON block include file, fdns01. An example of the fdns01 include file is shown below.

In this fdns01 example, the total grid size of the problem is IIQMAX = 110000, total number of wall function points is IWP = 11000; total number of grid points for the sliding boundary, ISLMAX, is set to be 1; maximum number of chemical elements is MEL = 10 (** do not change **); maximum number of chemical species is NSPM = 11; the multi-species thermodynamics data dimension, ISPAMAX, must be equal to IIQMAX to activate the multi-species option; the maximum number of particle trajectories, NPMAX, is set to be 1; the particle property dimension, IJKPMX, must be set to be IIQMAX in order to activate the Lagrangian particle tracking option; the maximum number of porous volumes is NPOROX = 1; the porosity dimension, IJKVMX, must be set to IIQMAX to activate the porosity option; the maximum number of surface porosity is IJKWMX = 1; MST is the maximum number of chemical reaction steps; MZON is the maximum number of blocks (zones); MBIF is the maximum number of zonal interfaces; MBIO is the maximum number of flow boundaries; and MBWA, MBSN and MZPO are the maximum numbers of wall segments, singularity lines and porosity zones in the input data file-- Afort.11", respectively.

The main program of the present code, which is the main driver for other subroutines, defines input/output units and control parameters, provides problem restart modifications (through include file fmain01), defines the solution calling sequence, and provides time-marching control and timely input of run-time problem modifications and data output (through include file fmain02). Input and output units, IR1, IR2, IR3, IW1, IW2, and IW3, are assigned in the main program. Unit IR1 (input data file fort.11) is for setting up the flow domain and problem control parameters. Flow domains sizes, zonal interfaces locations, boundaries locations and types, wall block locations, job control parameters, upwind scheme selections, turbulence model selections, setting the reference conditions
FDNS-RFV/PVM

(viscosity = 1/Re, Mach number, reference density, reference velocity, reference temperature, reference length, etc.), and thermodynamics and reaction data, etc., are included in unit IR1. Units IR2 and IR3 are assigned for restart grid and flow field data files (fort.12 and fort.13) respectively. The flow solution convergence history and evolution of the monitoring point flow variables are printed from unit 6 (fort.6 or nohup.out). IW1 or unit 21 is not used in the current version. Units IW2 and IW3 are used for the grid data and flow field solution output. The current version also provides PLOT3D format grid and flow outputs through file units fort.91, fort.92 and fort.93 for post processing. The file unit fort.91 contains the grid data, fort.92 contains five variables (i.e. density, u-velocity, v-velocity, w-velocity, and total pressure), and fort.93 also contains five flow variables (i.e. density, pressure, temperature, Mach number, and species 1 concentration). In addition, unit fort.94 contains both grid and flow outputs (including density, velocities, pressure, temperature and Mach numbers), to a commercial graphics software “TECPLOT” for purpose of post processing. Different variables output or different data format (a binary data option is given in the source code) can be obtained by modifying the subroutine DATAIO (only the PLOT3D section which is located near the end of DATAIO) in f1.f. The program start or restart status is defined by setting IDATA = 2 or IDATA = 1 in the input data (fort.11) respectively. When IDATA = 2 is selected, grid and initial flow field data must be made available for one the include files, fexmp01, for a fresh start.

Grid and initial flow field data files can be generated using other preprocessor type grid generation codes such as EAGLE and GENIE, etc., and user developed initial flow field generation codes. A grid/initial flow generator included in Appendix C provides an example of how these preprocessors can be constructed. The user may follow these examples for the generation of other new cases.

To present the flowfield solutions graphically using the output files of the FDNS-RFV/PVM code, graphics packages using DISSPLA, TEKTRONICS, VOGLE or open-GL utilities such as PLOT3D (developed at NASA/Ames Research Center) or a commercial software--TECPLOT can be used to plot the grid, velocity vectors, and contour lines of selected field quantities. Other flow parameter outputs such as surface pressure, skin friction distributions, and heat transfer coefficient distributions (which are output from the include file fmain02) can also be plotted.

Other important features of the present code are further elaborated upon in the following four sections. This introduction provides merely a general outline of the code. However, it is thought to be detailed enough for the FDNS-RFV/PVM users to perform daily engineering analysis applications.

3.1 CODE STRUCTURE

The FDNS-RFV/PVM code was designed for robustness and user friendliness. The entire program is written in standard FORTRAN 77 language. The code is generally not machine dependent. It has been successfully tested on the personal computers with Windows-95, Windows-NT and Linux operating systems, IBM RISC/6000 workstation, IBM 3084 main-frame computer, the Cray-XMP, Cray-2 and Cray-YMP supercomputers.

Besides the main program, there are 91 subroutines and three entries in the FDNS-RFV/PVM code. They are grouped in 7 files namely “fdns.f”, “f1.f”, “f2.f”, “f3.f”, “f4.f”, “f5.f”, and “f6.f”. In addition, there is a data base file, “dbase.dat”, which contains thermodynamics data of various species for the real-fluid EOS. The subroutines' names and their major functions are summarized
FDNS-RFV/PVM

below.

<In fdns.f>

FDNS  the main program which is the main driver for other subroutines, defines input/output units and control parameters, provides problem restart modifications (through include file fmain01), defines the solution calling sequence, and provides time-marching control and timely input of run-time problem modifications and data output (through include file fmain02).

EXAMP  to allow the user to generate (or read in) grid and prepare initial flow field data using an include file, fexmp01, for a fresh start.

XISENT  for calculating nozzle Mach number, pressure, density, and temperature variations (as a function of local versus throat area ratio) based on isentropic relations.

USUBIO  to calculate and store initial flow field total pressure, total enthalpy and mass flux conditions and provide inlet boundary conditions based on total conditions or mass conservation.

CHOEQ  to calculate equilibrium species concentrations for the wet-CO mechanism by solving a set of algebraic equations.

<In f1.f>

ZONCHK  to check the grid indices at the zonal interface, to calculate as well as exchange the grid spacing between the grid point at the zonal interface and its neighboring point, and also to compute the angle between cyclic boundaries.

CYCANG  to calculate the angle between each pair of cyclic boundaries.

BCCOND  to provide implicit and explicit boundary conditions for all flow variables that includes mass conservation conditions, pressure conditions and pressure equation boundary condition setting.

DATINN  to read in FDNS-RFV/PVM restart grid and flow field data files (IR2 - fort.12 and IR3 - fort.13).

DATOUT  to output files FDNS-RFV/PVM restart grid and flow field data files (IW2 - fort.22 and IW3 - fort.23).

INFACE  to perform multiple-block, zonal-interface interpolation boundary conditions that requires overlaid zonal interfaces. Interface identification, interface grid movement and interface interpolations are handled in this subroutine. Make sure that the statement ‘include ’iface.inc’ in INFACE is activated.
### INFINT
This is a tool for 2-D interpolations using bi-linear interpolation scheme.

### INIT
To provide initialization of problem control parameters, model constants and zero-out flow field variables (not for initial flow conditions).

### WALLFN
To calculate near wall velocity profiles, heat flux to the wall, static enthalpy on the wall, and near-wall turbulence quantities by using wall function models for turbulent flow boundary conditions. The heat of pyrolysis model for hybrid fuel regression is also evaluated here.

#### <In f2.f>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLVEU</td>
<td>To provide solutions for the momentum and energy equations using high-order upwind, TVD, or central difference schemes plus adaptive dissipation terms.</td>
</tr>
<tr>
<td>SOLVEQ</td>
<td>To provide solutions for the turbulence model transport equations.</td>
</tr>
<tr>
<td>SOLVES</td>
<td>To provide solutions for the chemical species mass fraction transport equations.</td>
</tr>
<tr>
<td>SOLVET</td>
<td>To provide solutions for the thermal conduction equation for block of solid wall points.</td>
</tr>
<tr>
<td>SOLVEP</td>
<td>To solve the pressure correction equation and perform pressure, temperature, velocity and density field updating.</td>
</tr>
</tbody>
</table>

#### <In f3.f>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AINDEX</td>
<td>To convert the global one-dimensional indices, IJK, into the multiple-zone three-dimensional indices (NZ, I, J, K).</td>
</tr>
<tr>
<td>AREAIO</td>
<td>To calculate density-weighted inlet and outlet areas.</td>
</tr>
<tr>
<td>BCCHAR</td>
<td>To provide outlet properties extrapolations and perform outlet and symmetry plane velocity vector conditions.</td>
</tr>
<tr>
<td>BOUNC</td>
<td>This is a driver for wall function points by calling WALLFN.</td>
</tr>
<tr>
<td>DIRCOS</td>
<td>To register wall element identification and wall function control parameters and calculate wall surface orientations.</td>
</tr>
<tr>
<td>FLOWIO</td>
<td>To calculate the mass flow rates at the outlet boundaries.</td>
</tr>
<tr>
<td>LINER0</td>
<td>To perform 1-D regular TDMA matrix inversion.</td>
</tr>
</tbody>
</table>
LINER1 to perform 1-D periodic TDMA matrix inversion.

LINERA to perform global point-by-point, L-U iterative, conjugate gradient, or GMRES matrix solution.

CGSOLV a driver for choosing either conjugate gradient or GMRES method.

PNTBYPNT point-by-point solver in the zone-by-zone matrix solver.

PTCR1 preconditioned zone-by-zone conjugate residual matrix solver with DKR and DD factorization.

PGMRES1 preconditioned zone-by-zone GMRES matrix solver.

PTCR2 preconditioned multi-zone conjugate residual matrix solver with DKR and DD factorization.

PGMRES2 preconditioned multi-zone GMRES matrix solver.

BNDCOEF to reset the link coefficients of the zonal boundary points matrix to zero.

DOTSUM to calculate inner product of two vector arrays in the zone-by-zone matrix solver.

MATRIX to perform multiplication of matrix in the zone-by-zone matrix solver.

PRINTN to print out a matrix.

DECOM1 Dupont-Kendall-Rachford (DKR) factorization in the zone-by-zone matrix solver.

DECOM2 double decomposition (DD) in the zone-by-zone matrix solver.

INVER1 inverse LDU for DKR decomposition in the zone-by-zone matrix solver.

INVER2 inverse LDU for DD decomposition in the zone-by-zone matrix solver.

PTBYPT point-by-point solver in the zone-by-zone matrix solver.

IJKXYZ to calculate total grid number in each zone.

DOTSUM2 to calculate inner product of two vector arrays in the multi-zone matrix solver.

MATRIX3 to perform multiplication of matrix in the multi-zone matrix solver.

DECOM3 Dupont-Kendall-Rachford (DKR) factorization in the multi-zone matrix solver.
DECOM4  double decomposition (DD) in the multi-zone matrix solver.
INVER3  inverse LDU for DKR decomposition in the multi-zone matrix solver.
INVER4  inverse LDU for DD decomposition in the multi-zone matrix solver.
PTBYPT2 point-by-point solver in the multi-zone matrix solver.
UPBCPT  to update values for boundary conditions and zonal interface points.
PRINTM  to print out a matrix field.
EQUAL1  to assign a constant to a vector array.
EQUAL2  to assign the values of one vector array to another vector array.
DIVID1  to divide a vector array with a constant.
NORM2  to calculate the norm of a vector array.
RVA4    to assign 4 real constants.
LINKFA  to identify the indices for the grid points at the zonal interface boundaries.

<In f4.f>

SOURCE to evaluate source terms for all transport equations. Multiple-phase inter-phase source terms are also calculated here.

SOURCXM 3 entries are included:
NEWVIS --- to calculate turbulence eddy viscosity.
BBLOCK --- to assign wall element identifications.
PROPTY --- to calculate thermodynamics data and chemical reaction source terms (use include file propty.inc)

TRANF    to calculate the Jacobian of the coordinate transformation and grid spacing variations.

UNEWIO   to perform outlet velocity corrections based on mass conservation conditions.

UVCON    this subroutine is currently null.

WALVAL   for assigning boundary values for the wall surface points, inlet planes, outlet planes, symmetry planes and singularity lines.

HEAT00 a general fluid thermodynamics subroutines driver which calls HEAT2A,
HEAT2C, HEAT2D, FLINT, NBSLOX and NBSH2O.

HEAT2A to calculate the thermodynamics properties and to assemble finite-rate chemistry source terms.

SOOTOX to calculate the reaction rate for the soot oxidation chemistry model.

HEAT2B to find ideal gas temperature (in °K) based on the given enthalpy and gas species concentrations using Newton's method.

GAUSS to solve a matrix by using Gaussian elimination with the pivoting method.

CPHG to calculate specific heat (C_p), enthalpy (h/R), and Gibbs free energy at a given temperature based on CEC thermal data.

HEAT2C to calculate pressure and enthalpy based on the density and temperature for either ideal gas or real fluid.

HEAT2D to find ideal gas temperature (in °K) based on the given enthalpy and gas species concentrations by using Newton's method, and calculate C_p and γ.

NBSLOX to provide table look-up of LOX properties (H-P-T diagram) based on the National Bureau of Standard data.

NBSH2O to provide table look-up of water properties (H-P-T diagram) for pressure less than 1.5 atm.

<In f5.f>

LPTSD this is a main driver for setting up the particulate phase integration scheme that includes the particle initial conditions, Lagrangian integration of the particle trajectories and the assembly of the inter-phase source terms.

HPTDAT to calculate temperatures of AL2O3 particles based on their enthalpy.

SEARCH to locate particle inside the computational domain. A multiple zone algorithm is included.

PSOURC to calculate the drag forces and heat transfer rates for the particle based on the inter-phase slip conditions.

POROST this is a main driver for a porosity model.

DRAGHT to calculate the drag forces and heat transfer based on the assigned porosity.

<In f6.f>
FDNS-RFV/PVM

FLINT the interface with real fluids model.

ACENF to calculate acentric factor using Lee-Kesler model.

DBASE routine containing data base for fluids thermodynamics properties (see file "dbase.dat" for species represented).

IDMIX control routine for ideal mixture using HBMS equation of state.
mode=
  1: initialization
  2: solve pressure, enthalpy based on density & temperature
  3: solve density, enthalpy based on pressure & temperature
  4: solve temperature, enthalpy based on density & pressure
  5: solve pressure, temperature based on density & enthalpy
  6: solve density, temperature based on pressure & enthalpy

HBMS routine to compute real fluid properties using HBMS EOS.

SATLINE to compute saturation line conditions.

THERMAL to calculate ideal gas thermodynamic properties.

VFROMPT routine to determine volume from given pressure and temperature.

LAGRAN Lagrange interpolation module.

<In zone.f>

NETINIT to read in the hostname of the machine where the employed processors reside, and then start up the executable of the task on the slave machine(s) if the parallel computing option is selected, also to read in the destination of all input data files as well as the FDNS-RFV/PVM executable.

ZONEDIS to exchange the grid distance next to the zonal interface of a given zone with its neighboring zone.

ZONEBC to exchange the numerical solutions at the zonal interface of a given zone with its neighboring zone.

PACKBC to pack the numerical solutions at the zonal interface into a vector array to be transmitted by the PVM routines.

DISTAN to calculate the grid distance between the grid point at the zonal interface and its neighboring grid point.

WALLMW to set the flag of identifying grid points of wall blocks at the zonal interface.
IJKBC to set the index of the vector array used to pack the numerical solutions at the zonal interface

GETHOSTS to read in the hostname of the machine where the employed processors reside, and read in the destination (path to the directory) of all input data files as well as the FDNS-RFV/PVM executable.

IF_NUMBER to identify whether all elements of a string are numbers or not.

<In io.f>

READ_RESTRAT_RFV to read in FDNS (RFV version format) restart grid and flow files from the working directory of each host machine.

WRITE_RESTART_RFV to print out FDNS (RFV version format) restart grid and flow files to the working directory of each host machine.

READ_GRID_RFV to read in FDNS (RFV version format) grid file.

READ_FLOW_RFV to read in FDNS (RFV version format) flow file.

WRITE_GRID_RFV to print out FDNS (RFV version format) grid file.

WRITE_FLOW_RFV to print out FDNS (RFV version format) flow file.

CHECK_IIQMAX to check whether the total number of grid points exceeds the maximum memory allocation or not.

READ_PLOT3DG to read in PLOT3D grid file.

WRITE_PLOT3DG to print out PLOT3D grid file.

READ_PLOT3DQ to read in PLOT3D flow file.

WRITE_PLOT3DQ to print out PLOT3D flow file.

<In pvm.f>

INITPVM to start up the child process of the parallel computing from the master host by using the PVM SPMD model.

NETCAST to pack the data of a real variable into the message buffer and broadcast it to all processors.

NETRECV to receive a message buffer sent by a specific processor and unpack it into data of a real variable.
NETCAST_IV_N to pack an ‘n’-element array of integers into the message buffer and broadcast it to all processors.

NETRECV_IV_N to receive a message buffer sent by a specific processor and unpack it into an ‘n’-elements array of integers.

NETCAST_RV_N to pack an ‘n’-element array of real numbers into the message buffer and broadcast it to all processors.

NETRECV_RV_N to receive a message buffer sent by a specific processor and unpack it into an ‘n’-element array of real numbers.

NETCAST_STRING to pack an ‘n’-element array of character strings into the message buffer and broadcast it to all processors.

NETRECV_STRING to receive a message buffer sent by a specific processor and unpack it into an ‘n’-element array of character strings.

SEND_STRING to pack an ‘n’-element array of character strings into the message buffer and send it to a specific processor ID.

RECV_STRING to receive a message buffer sent from a specific processor ID and unpack it into an ‘n’-element array of character strings.

CHAR_ICHAR to convert a character string to an integer array.

ICHAR_CHAR to convert an integer array to a character string.

SENDIT1 to pack one integer number into the message buffer and send it to a specific processor ID.

RECVIT1 to receive a message buffer sent from a specific processor ID and unpack it to get one integer number.

SENDIT2 to pack two integer numbers into the message buffer and send it to a specific processor ID.

RECVIT2 to receive a message buffer sent from a specific processor ID and unpack it to get two integer numbers.

SENDIT3 to pack three integer numbers into the message buffer and send it to a specific processor ID.

RECVIT3 to receive a message buffer sent from a specific processor ID and unpack it to get three integer numbers.

SENDIT4 to pack four integer numbers into the message buffer and send it to
aspecific processor ID.

**RECVIT4** to receive a message buffer sent from a specific processor ID and unpack it to get four integer numbers.

**SENDNB1** to pack one real number into the message buffer and send it to a specific processor ID.

**RECVNB1** to receive a message buffer sent from a specific processor ID and unpack it to get one real number.

**SENDNB2** to pack two real numbers into the message buffer and send it to a specific processor ID.

**RECVNB2** to receive a message buffer sent from a specific processor ID and unpack it to get two real numbers.

**SENDNB3** to pack three real numbers into the message buffer and send it to a specific processor ID.

**RECVNB3** to receive a message buffer sent from a specific processor ID and unpack it to get three real numbers.

**SENDNB4** to pack four real numbers into the message buffer and send it to a specific processor ID.

**RECVNB4** to receive a message buffer sent from a specific processor ID and unpack it to get four real numbers.

**SENDIT** to pack an ‘n’-element array of integer numbers starting at a given index into the message buffer and send it to a specific processor ID.

**RECVIT** to receive a message buffer sent from a specific processor ID and unpack it to get an ‘n’-element array of integer numbers starting at a given index.

**SENDNB** to pack an ‘n’-element array of real numbers starting at a given index into the message buffer and send it to a specific processor ID.

**RECVNB** to receive a message buffer sent from a specific processor ID and unpack it to get an ‘n’-element array of real numbers starting at a given index.

**NETSUM** to sum up the real variable sent from all processors and broadcast the result to all processors.

**EXITNET** to terminate the present process from PVM.
<In flib.f>

GETWORD  to convert an input character string into a character array without blank space.

GETNUM   to get the processor index from an input line.

FIND_WORD to find the key word for an entry in the input data file.

INUMBER_CNUMBER to convert a number character to an integer.

LENGTH   to obtain the length of a character string not including the blank space.

READCARD1 to read in a dummy line from the input data file.

IVA4     to assign 4 integer constants.
The basic structure of the FDNS-RFV/PVM code is depicted in Chart 3.1 which is a flow chart in the main program.

Chart 3.1 Flow Chart of the FDNS-RFV/PVM Main Program
Chart 3.1 Flow Chart of the FDNS-RFV/PVM Main Program, continued.

- Start Time Marching
- LPTSD(2), if activated
- POROST(2), if activated
- SOLVES, if activated
  - SOLVEU
  - SOLVET, if activated
  - SOLVEQ, if activated
  - SOLVEP
  - PROPTY, if activated
    - Print out Residuals, if yes
      - NEWVIS, if activated
        - USUBIO(2)
          - reset inflow conditions based on users’ option
            - DATOUT, if yes
              - print out restart file and post-processing data files
                - Converged?
                  - yes
                    - EXITNET
                      - Stop
                  - no
### 3.2 INPUT DATA (FORT.11) DEFINITION

The input data file (fort.11) of the FDNS-RFV/PVM code consists of 19 card groups and 2 entries. Definitions of these input data are given below. A sample input file is listed in Appendix A.

<table>
<thead>
<tr>
<th>Card Group #1</th>
<th>Defines the case title and whether the problem is 2-D or 3-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format (1 line)</td>
<td>Title (Put title of the problem here -- maximum 60 characters)</td>
</tr>
<tr>
<td>Format (1 line)</td>
<td>IDIM,</td>
</tr>
<tr>
<td>Definition</td>
<td>IDIM = 2: for 2-D planar or axisymmetric flow problems</td>
</tr>
<tr>
<td></td>
<td>IDIM = 3: for 3-dimensional flow problems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Card Group #2</th>
<th>Specifies zonal information and number of flow and wall boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format (1 line)</td>
<td>IZON, IZFACE, IBND, ID, ISNGL,</td>
</tr>
<tr>
<td>Definition</td>
<td>IZON: number of zones or mesh blocks</td>
</tr>
<tr>
<td></td>
<td>IZFACE: number of patched interfaces</td>
</tr>
<tr>
<td></td>
<td>IBND: number of flow boundaries (e.g. inlet, outlet or symmetry planes)</td>
</tr>
<tr>
<td></td>
<td>ID: number of wall elements (blocks)</td>
</tr>
<tr>
<td></td>
<td>ISNGL: number of singularity lines/surfaces</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Card Group #3</th>
<th>Specifies zonal grid size and zonal rotational/translational speeds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format (1 line*IZON)</td>
<td>IZT, JZT, KZT, LPROC, CBG1, CBG2, CBG3, CBV1, CBV2, CBV3</td>
</tr>
<tr>
<td>Definition</td>
<td>IZT(II): I-max in zone II</td>
</tr>
<tr>
<td></td>
<td>JZT(II): J-max in zone II</td>
</tr>
<tr>
<td></td>
<td>KZT(II): K-max in zone II</td>
</tr>
<tr>
<td></td>
<td>LPROC(II): index of the processor which handles zone II</td>
</tr>
<tr>
<td></td>
<td>CBG1(II): coordinate rotation speed (X_{ref}\Omega_x/U_{ref}) of zone II about X-axis</td>
</tr>
<tr>
<td></td>
<td>CBG2(II): coordinate rotation speed (X_{ref}\Omega_y/U_{ref}) of zone II about Y-axis</td>
</tr>
<tr>
<td></td>
<td>CBG3(II): coordinate rotation speed (X_{ref}\Omega_z/U_{ref}) of zone II about Z-axis</td>
</tr>
<tr>
<td></td>
<td>CBV1(II): coordinate translation speed of zone II in X-direction</td>
</tr>
<tr>
<td></td>
<td>CBV2(II): coordinate translation speed of zone II in Y-direction</td>
</tr>
<tr>
<td></td>
<td>CBV3(II): coordinate translation speed of zone II in Z-direction</td>
</tr>
</tbody>
</table>
Card Group #4 | Identifies the zonal interface matching indices
---|---
**Format** | IFCYC, IZB1, IZF1, IJZ11, IJZ12, JKZ11, JKZ12, INONUF
| IZB2, IZF2, IJZ21, IJZ22, JKZ21, JKZ22,
**Definition** | **IFCYC:**

- $> 0$, IZFACE counter (not used in the code)
- $= -1$, cyclic boundaries about I-axis
- $= -2$, cyclic boundaries about J-axis (not ready)
- $= -3$, cyclic boundaries about K-axis (not ready)

**IZB1:** zonal index of interface plane #1

**IZF1:**

- $= 1$: I=max or East boundary
- $= 2$: I=1 or West boundary
- $= 3$: J=max or North boundary
- $= 4$: J=1 or South boundary
- $= 5$: K=max or Top boundary
- $= 6$: K=1 or Bottom boundary

**IZB2:** zonal index of interface plane #2

**IZF2:** interface plane identifier for plane #2

**IJZ11, IJZ12:** the starting and ending points of the first running index on the interface plane #1

**JKZ11, JKZ12:** the starting and ending points of the second running index on the interface plane #1

**IJZ21, IJZ22:** the starting and ending points of the first running index on the interface plane #2

**JKZ21, JKZ22:** the starting and ending points of the second running index on the interface plane #2

**INONUF:**

- $= 0$, for matched grids at the zonal interface
- $> 0$ and $< 50$, for non-matching grids at the zonal interface (to be implemented in the future)

**Example:** If IZF1 or IZF2 is either 1 or 2 then IJZ11, IJZ12, IJZ21 and IJZ22 are the indices in J-direction, and JKZ11, JKZ12, JKZ21 and JKZ22 are the indices in K-direction.

If IZF1 or IZF2 is either 3 or 4 then IJZ11, IJZ12, IJZ21 and IJZ22 are the indices in I-direction, and JKZ11, JKZ12, JKZ21 and JKZ22 are the indices in K-direction.

If IZF1 or IZF2 is either 5 or 6 then IJZ11, IJZ12, IJZ21 and IJZ22 are the indices in I-direction, and JKZ11, JKZ12, JKZ21 and JKZ22 are the indices in J-direction.

**Note:** The interface patching surface indices for planes #1 and #2 (i.e. IJZ11→IJZ12 and IJZ21→IJZ22, JKZ11→JKZ12 and JKZ21→JKZ22 must have consistent running order).

Also, IJZ12 > IJZ11 and JKZ12 > JKZ11 (but not necessary for IJZ21, IJZ22, JKZ21, and JKZ22).
<table>
<thead>
<tr>
<th>Card Group #5</th>
<th>Specifies flow boundaries (inlet, outlet, symmetry)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>IBCZON, IDBC, ITYBC, IJBB, IJBS, IJBT, IKBS, IKBT,</td>
</tr>
<tr>
<td>(1 line*IBND)</td>
<td></td>
</tr>
<tr>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>IBCZON</td>
<td>zonal index for the flow boundary</td>
</tr>
<tr>
<td>IDBC</td>
<td>boundary facing index:</td>
</tr>
<tr>
<td></td>
<td>= 1: I=I-max or East</td>
</tr>
<tr>
<td></td>
<td>= 2: I=1 or West</td>
</tr>
<tr>
<td></td>
<td>= 3: J=J-max or North</td>
</tr>
<tr>
<td></td>
<td>= 4: J=1 or South</td>
</tr>
<tr>
<td></td>
<td>= 5: K=K-max or Top</td>
</tr>
<tr>
<td></td>
<td>= 6: K=1 or Bottom</td>
</tr>
<tr>
<td>ITYBC</td>
<td>identifies boundary type:</td>
</tr>
<tr>
<td></td>
<td>= -2: inlet with mass flow rate and velocities fixed</td>
</tr>
<tr>
<td></td>
<td>= -1: inlet with mass flow rates conserved (e.g. solid fuel blowing surfaces)</td>
</tr>
<tr>
<td></td>
<td>= 0: inlet with all variables fixed (e.g. supersonic)</td>
</tr>
<tr>
<td></td>
<td>= 1: inlet with constant total pressure (compressible flow only)</td>
</tr>
<tr>
<td></td>
<td>= 2: outlet boundary</td>
</tr>
<tr>
<td></td>
<td>= 3: symmetry plane (can also be regarded as slip/inviscid wall boundary conditions, but should not be combined with wall block in Card Group #6)</td>
</tr>
<tr>
<td>IJBB</td>
<td>I, J or K location (depends on IDBC) of the boundary</td>
</tr>
<tr>
<td>IJBS, IJBT</td>
<td>boundary starting and ending indices (for I or J)</td>
</tr>
<tr>
<td>JKBS, JKBT</td>
<td>boundary starting and ending indices (for J or K)</td>
</tr>
</tbody>
</table>
Card Group #6 | Specifies wall block indices
---|---
Format (1 line*ID) | IWBZON, L1, L2, M1, M2, N1, N2, IWTM, HQDOX, IWALL, DENNX, VISWX.
Definition | IWBZON: zonal index for the wall block
 | L1, L2: starting and ending indices in the I-direction
 | M1, M2: starting and ending indices in the J-direction
 | N1, N2: starting and ending indices in the K-direction
 | IWTM: solid-wall thermal boundary condition options
 | = -1: for fixed temperature wall boundary
 | = 1: for heat-flux (= HQDOX) wall boundary
 | HQDOX: non-dimensional wall heat flux when IWTM=1, the value is positive if it is from wall to fluid
 | Normalization for $\dot{Q}$:
 | SI Units = $\dot{Q}/(\rho_{ref}U_{ref}C_{ref}T_{ref})$
 | English Units = $\dot{Q}/(32.174\rho_{ref}U_{ref}C_{ref}T_{ref})$
 | IWALL: solid wall heat conduction option
 | = 0: to deactivate
 | = 1: to activate
 | DENNX: non-dimensional solid wall density (wall density/$\rho_{ref}$)
 | VISWX: non-dimensional solid wall thermal conductivity
 | = $\kappa/(X_{ref}\rho_{ref}U_{ref}C_{ref})$

Note: The specified values of DENNX and VISWX will be meaningful only when IWALL = 1 is selected, and the program will set IWTM = -1, since this is a correct combination.
## Card Group #7

Specifies the singularity lines

<table>
<thead>
<tr>
<th>Format (1 line*ISNGL)</th>
<th>ISNZON, ISNBC, ISNAX, ISNBS, ISNBT,</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ISNZON</th>
<th>Zonal index for the singularity lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISNBC</td>
<td>Singularity line boundary facing index</td>
</tr>
<tr>
<td></td>
<td>= 1: I=I-max or East</td>
</tr>
<tr>
<td></td>
<td>= 2: I=1 or West</td>
</tr>
<tr>
<td></td>
<td>= 3: J=J-max or North</td>
</tr>
<tr>
<td></td>
<td>= 4: J=1 or South</td>
</tr>
<tr>
<td></td>
<td>= 5: K=K-max or Top</td>
</tr>
<tr>
<td></td>
<td>= 6: K=1 or Bottom</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISNAX</th>
<th>Orientation of the singularity line axis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>on I-J plane (ISNBC=5 or 6)</td>
</tr>
<tr>
<td></td>
<td>ISNAX = 1 for I-axis</td>
</tr>
<tr>
<td></td>
<td>ISNAX = 2 for J-axis</td>
</tr>
<tr>
<td></td>
<td>on J-K plane (ISNBC = 1 or 2)</td>
</tr>
<tr>
<td></td>
<td>ISNAX = 1 for J-axis</td>
</tr>
<tr>
<td></td>
<td>ISNAX = 2 for K-axis</td>
</tr>
<tr>
<td></td>
<td>on K-I plane (ISNBC = 3 or 4)</td>
</tr>
<tr>
<td></td>
<td>ISNAX = 1 for I-axis</td>
</tr>
<tr>
<td></td>
<td>ISNAX = 2 for K-axis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition</th>
<th>ISNBS, ISNBT</th>
<th>Starting and ending indices along ISNAX</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ISNBS</th>
<th>ISNBT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Card Group #8</th>
<th>I/O parameters and problem control parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format (1 line)</td>
<td>IDATA, IGE0, ITT, ITPNT, ICOUP, NLIMT, IAX, ICYC,</td>
</tr>
<tr>
<td>Definition</td>
<td>IDATA restart options</td>
</tr>
<tr>
<td></td>
<td>= 1: for regular restart runs. Restart grid and flow files, fort.12 and fort.13, must be made available. The format of the restart files can be specified by IOFINN in Card Group #15</td>
</tr>
<tr>
<td></td>
<td>= 2: for example start run. Users must implement the subroutine EXAMP in the fexmp01 include file to properly read in the pre-generated grid and flow data.</td>
</tr>
<tr>
<td>IGEO geometry parameter (for user applications)</td>
<td></td>
</tr>
<tr>
<td>= 1: is specifically for problems without inlets and outlets (e.g. cavity flows)</td>
<td></td>
</tr>
<tr>
<td>= 19: is reserved for linear cascades applications</td>
<td></td>
</tr>
<tr>
<td>ITT number of time steps limit</td>
<td></td>
</tr>
<tr>
<td>ITPNT the frequency on printing out solutions (through files fort.22, fort.23, fort.91, fort.92, fort.93, and fort.94)</td>
<td></td>
</tr>
<tr>
<td>ICOUP number of pressure correctors (typically 1 for steady-state applications and 3-6 for transient or rough initial start applications)</td>
<td></td>
</tr>
<tr>
<td>NLIMT = 1: for regular run</td>
<td></td>
</tr>
<tr>
<td>= 0: for printing out the initial or restart files without going through solution procedures</td>
<td></td>
</tr>
<tr>
<td>IAX = 1: for 2-D planar or 3-D flows</td>
<td></td>
</tr>
<tr>
<td>= 2: for 2-D axisymmetric flow problems</td>
<td></td>
</tr>
<tr>
<td>ICYC cyclic or periodic boundary conditions identifier. Currently, only ICYC = 3 (in K-direction) is active for turbomachinery applications, where all flow variables at K=1 are the same as those at K=K-max in the corresponding zones.</td>
<td></td>
</tr>
<tr>
<td>Card Group #9</td>
<td>Time-step size, upwind schemes and time-marching scheme selections.</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>Format (1 line)</td>
<td>DTT, IREC, REC, THETA, BETAP, IEXX, PRAT,</td>
</tr>
<tr>
<td>Definition</td>
<td>DTT non-dimensional time step size, $\Delta t (U_{ref}/X_{ref})$</td>
</tr>
<tr>
<td></td>
<td>IREC selects upwind scheme options</td>
</tr>
<tr>
<td></td>
<td>$=0$: for second-order upwind scheme</td>
</tr>
<tr>
<td></td>
<td>$=1$: for third-order upwind scheme</td>
</tr>
<tr>
<td></td>
<td>$=2$: for second-order central scheme</td>
</tr>
<tr>
<td></td>
<td>$=3$: for second-order TVD scheme</td>
</tr>
<tr>
<td></td>
<td>$=4$: for third-order TVD scheme</td>
</tr>
<tr>
<td></td>
<td>REC upwind damping parameter (0.1 recommended)</td>
</tr>
<tr>
<td></td>
<td>$=0.0$: for second-order accuracy</td>
</tr>
<tr>
<td></td>
<td>$=1.0$: for first-order upwind scheme</td>
</tr>
<tr>
<td></td>
<td>THETA time-marching scheme parameter</td>
</tr>
<tr>
<td></td>
<td>$=1.0$: for steady-state applications</td>
</tr>
<tr>
<td></td>
<td>$=0.99$: for implicit-Euler transient applications</td>
</tr>
<tr>
<td></td>
<td>$=0.5$: for Crank-Nicholson second-order accurate transient applications</td>
</tr>
<tr>
<td></td>
<td>BETAP $\leq 1.0$ pressure updating under-relaxation parameter, typically 1.0; small values can be used to reduce the amount on pressure corrections for rough start initial runs (in this case, choose ISWP $\geq 80$ as explained in Card Group #12)</td>
</tr>
<tr>
<td></td>
<td>$&gt;1.0$ factor for the diagonal term of matrix coefficients in the pressure correction equation to maintain stability of matrix solver (typically 1.01 for incompressible flows, single species or pre-mixed multi-species flows)</td>
</tr>
<tr>
<td></td>
<td>IEXX Outlet extrapolation parameter for scalar quantities</td>
</tr>
<tr>
<td></td>
<td>$=1$: for zero-gradient extrapolation</td>
</tr>
<tr>
<td></td>
<td>$=2$: for linear extrapolation</td>
</tr>
<tr>
<td></td>
<td>PRAT specifies outlet boundary condition options</td>
</tr>
<tr>
<td></td>
<td>$=-1.$: for supersonic outlet B.C.</td>
</tr>
<tr>
<td></td>
<td>$=0.0$: for outlet mass conservation B.C.</td>
</tr>
<tr>
<td></td>
<td>$&gt;0.$: for outlet fix pressure b.c. The outlet pressure reference point (IPEX, JPEX) is used here. Pressure at this point is maintained at a value of $PRAT*PPCN$, where PPCN is the atmospheric pressure (1 atm).</td>
</tr>
</tbody>
</table>
Card Group #10 | Specifies inlet, outlet pressure points and data monitoring point
---|---
**Format (1 line)** | IPC, JPC, IPEX, JPEX, IMN, JMN,
**Definition** | IPC, JPC: flowfield reference point at the grid index of IPC in zone JPC (not the global grid index)
IPEX, JPEX: outlet pressure reference point (same way of indexing as IPC, JPC)
IMN, JMN: solution monitoring point

Card Group #11 | Gives reference viscosity, Mach number and options of turbulence models
---|---
**Format (1 line)** | VISC, IG, ITURB, AMC, GAMA, CBE, CBH, EREXT
**Definition** | VISC: dimensional fluid viscosity of air at the sea level (the unit has to be consistent with that specified by IUNIT)
IG = 1: for laminar flow option
IG = 2: for turbulent flow option
ITURB for turbulence model selection
ITURB = 1: for standard high-Re k-ε model
ITURB = 2: for extended high-Re k-ε model
ITURB = 3: for Lam-Bremhorst low-Re k-ε model
ITURB = 4: for H-G low-Re k-ε model
ICOMP = 0: for incompressible flow calculation
ICOMP = 1: for compressible flow calculation
GAMA: reference specific heat ratio (not used in the code)
CBE = 0: no buoyancy effect
CBE > 0.5: include buoyancy effect for compressible flow
CBE < 0: non-dimensional buoyancy force parameter for incompressible flow, |CBE| = Gr/Re², where Gr is the Grashof number and Re is the Reynolds number
CBH select compressibility corrections for the k-ε turbulence model
CBH = 0.0: no compressibility correction
CBH = -1.0: for k-corrected model
CBH = -2.0: for ε-corrected model
CBH < -3.0: for T-corrected model where C_3(T/T_a)^{\gamma}, \lambda = *CBH*-3
EREXIT convergence criterion (typically 5.0E-05 for steady-state solutions)
<table>
<thead>
<tr>
<th>Card Group #12</th>
<th>Specifies number of zonal iterations in the matrix solver when (INFACE &gt; 0) is used for overlaid grid zonal interface interpolations and indicates orthogonal or non-orthogonal grid options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format (1 line)</td>
<td>ISWU, ISWP, ISWK, ISKEW,</td>
</tr>
</tbody>
</table>
| Definition | ISWU for the momentum and energy equations  
| | ≤ 80 number of iterations for the overlaid zonal boundaries by using point-by-point matrix solver  
| | > 80 conjugate gradient matrix solver (solving multi-zones as a whole) is used to solve the matrices until its residuals drop (ISWU-80) orders  
| | > 85 GMRES matrix solver (solving multi-zones as a whole) is used to solve the matrices until its residuals drop (ISWU-85) orders  
| | > 90 conjugate gradient matrix solver (solving zone-by-zone) is used to solve the matrices until its residuals drop (ISWU-90) orders  
| | > 95 GMRES matrix solver (solving zone-by-zone) is used to solve the matrices until its residuals drop (ISWU-95) orders  
| ISWP | for the pressure correction equations  
| | ≤ 80 number of iterations for the overlaid zonal boundaries by using point-by-point matrix solver  
| | > 80 Same as above for ISWU  
| ISWK | for the scalar equations (e.g. k, ε, and species equations)  
| | ≤ 80 number of iterations for the overlaid zonal boundaries by using point-by-point matrix solver  
| | > 80 Same as above for ISWU  
| ISKEW | non-orthogonal grid viscous flux option  
| | = 0: for orthogonal grid  
<p>| | = 1: for non-orthogonal grid |</p>
<table>
<thead>
<tr>
<th>Card Group #13</th>
<th>Specifies which equations are to be solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format (1 line)</td>
<td>INSO(IEQ): U, V, W, TM, DK, DE, FL, 8, EQ, VS, FM, SP,</td>
</tr>
<tr>
<td>Definition (0 to deactivate; 1 to activate)</td>
<td></td>
</tr>
<tr>
<td>U, V, W = 1:</td>
<td>solving the momentum equations</td>
</tr>
<tr>
<td>TM = 1:</td>
<td>solving the energy equation</td>
</tr>
<tr>
<td>DK, DE = 1:</td>
<td>solving the turbulence model</td>
</tr>
<tr>
<td>FL = 0:</td>
<td>for ideal gas flow model</td>
</tr>
<tr>
<td>= n &gt; 0:</td>
<td>for real fluid flow model with the quality of the n-th species being saved (In this case, a “FLUID” entry at the end of the input file is needed to identify the species which properties will be calculated through real fluid model.)</td>
</tr>
<tr>
<td>8</td>
<td>not used</td>
</tr>
<tr>
<td>EQ = 0:</td>
<td>no equilibrium chemistry</td>
</tr>
<tr>
<td>= 1:</td>
<td>H₂/O₂ equilibrium chemistry</td>
</tr>
<tr>
<td>= 2:</td>
<td>wet-CO equilibrium chemistry</td>
</tr>
<tr>
<td>VS = 1:</td>
<td>for updating the turbulence eddy viscosity</td>
</tr>
<tr>
<td>FM = 0:</td>
<td>de-activate the species mass-fraction equations (e.g. single species or air flows)</td>
</tr>
<tr>
<td>= 1:</td>
<td>activate the species mass-fraction equations</td>
</tr>
<tr>
<td>SP</td>
<td>for calculating the gas thermal properties, and selecting various treatment for species production term</td>
</tr>
<tr>
<td>= 1:</td>
<td>explicit chemistry model (penalty function)</td>
</tr>
<tr>
<td>= 11 or 12:</td>
<td>implicit chemistry model (1&lt;sup&gt;st&lt;/sup&gt; or 2&lt;sup&gt;nd&lt;/sup&gt;-order) with pseudo-time step size</td>
</tr>
<tr>
<td>= 21 or 22:</td>
<td>implicit chemistry model (1&lt;sup&gt;st&lt;/sup&gt; or 2&lt;sup&gt;nd&lt;/sup&gt;-order) with real time step size</td>
</tr>
<tr>
<td>= 31 or 32:</td>
<td>1&lt;sup&gt;st&lt;/sup&gt; or 2&lt;sup&gt;nd&lt;/sup&gt;-order implicit chemistry model with time integration (constant T, P)</td>
</tr>
<tr>
<td>= 33:</td>
<td>4&lt;sup&gt;th&lt;/sup&gt;-order PARASOL chemistry model with time integration (constant T, P)</td>
</tr>
<tr>
<td>= 41 or 42:</td>
<td>1&lt;sup&gt;st&lt;/sup&gt; or 2&lt;sup&gt;nd&lt;/sup&gt;-order implicit chemistry model with time integration (constant H, P)</td>
</tr>
<tr>
<td>= 43:</td>
<td>4&lt;sup&gt;th&lt;/sup&gt;-order PARASOL chemistry model with time integration (constant H, P)</td>
</tr>
</tbody>
</table>
Card Group #14 | Specifies number of gas species and reactions, and gives the reference conditions
---|---
Format (1 line) | NGAS, NREACT, IUNIT, DENREF, UREF, TREF, XREF,
Definition | NGAS number of chemical species which thermal properties in CEC tables will be read in
| = 0: for single species, ideal gas flow
| > 0: for multiple chemical species flow
| =-1: for LOX flow calculation where its thermodynamics properties are calculated from NBS table look-up
| =-3: for water flow calculation where its thermodynamics properties are calculated from table look-up
NREACT number on reaction steps to be used
| = 0: for non-reacting flow
| > 0: for finite-rate reacting flow
IUNIT = 1: for SI-unit reference conditions
| = 2: for English-unit reference conditions
DENREF reference density (in kg/m$^3$ or slug/ft$^3$), not used in the code
UREF reference velocity (in m/sec or ft/sec)
TREF reference temperature (in °K or °R), not used in the code
XREF reference length (in m or ft)
PREF reference pressure (in psi or N/m$^2$)

Card Group #15 | specify the format for input and output files
---|---
Format (1 line) | IGDINN, IOFINN, IOFOUT, IOFP3D
Definition | IGDINN = 1, unformatted PLOT3D grid file
| = 2, formatted PLOT3D grid file
IOFINN = 1, unformatted input grid and flow files
| = 2, formatted input grid and flow files
IOFOUT = 0, do not output grid and flow files
| = 1, unformatted output grid and flow files
| = 2, formatted output grid and flow files
IOFP3D = 0, do not output PLOT3D flow file
| = 1, unformatted output PLOT3D flow file
| = 2, formatted output PLOT3D flow file
### Card Group #16
If NGAS > 0, include the CEC thermodynamics data here

<table>
<thead>
<tr>
<th>Format</th>
<th>SPECIE, WTMOLE, HF(7,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4 lines*NGAS)</td>
<td>&lt;= (1 line)</td>
</tr>
<tr>
<td></td>
<td>HF(7,2) &lt;= (3 lines)</td>
</tr>
</tbody>
</table>

**Definition**
- **SPECIE**: name of the chemical species (20 characters)
- **WTMOLE**: molecular weight of the chemical species
- **HF(7,2)**: polynomial coefficients of CEC thermodynamics data of the species

### Card Group #17
If NREACT > 0, specifies the finite-rate reaction steps

<table>
<thead>
<tr>
<th>Format</th>
<th>REACTION: species names, N = 1,NGAS (only 1 line as a title)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2 or 3 lines*NREACT)</td>
<td>IREACT, A, B, E/RT, ITHIRD, IGLOB,</td>
</tr>
<tr>
<td></td>
<td>(STOCEF(N, IREACT), N=1,NGAS),</td>
</tr>
<tr>
<td></td>
<td>(STOCEG(N, IREACT), N=1,NGAS)**</td>
</tr>
</tbody>
</table>

**Definition**
- **IREACT**: reaction step counter
- **A**: reaction rate leading constant  
  (A=0 is designated for the soot-oxidation chemistry)
- **B**: reaction rate temperature exponent
- **E/R**: reaction rate activation energy constant  
  (if A=0, then E/R is the assumed diameter of the soot particle in meter, typically 4x10^-6 m)
- **ITHIRD**: third-body reaction indicator  
  - 0: deactivated  
  - n: for using the N-th species as third-body  
  - 999: for global (every species) third-body
- **IGLOB**: global reaction model indicator  
  - 0: elementary reactions with the rate of backward reaction is calculated from equilibrium constant and forward reaction rate  
  - 1: one-way reaction (either forward or backward reaction controlled by the sign of STCOEF); need only one input line of STCOEF  
  - 2: one-way reaction with power dependency; need input line for STCOEF and STCOEG
- **STCOEF**: stoichiometric coefficients of elementary reactions (negative signs apply to reactants and positive signs are for the products)
- **STCOEG**: power dependency coefficients
If IJKPMX=IIQMAX in the parameter specification, then read in the
following particle input control

<table>
<thead>
<tr>
<th>Card Group #18</th>
<th>Format (1 line)</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IDPTCL, IPREAD</td>
<td>IDPTCL</td>
</tr>
</tbody>
</table>
|                |                | number on particle initial condition input lines
|                |                | = 0: to deactivate particulate phase calculation
|                |                | = 1: to activate particulate phase calculation
|                |                | IPREAD     |
|                |                | = 0: read in particle inlet conditions in next card group
|                |                | = 1: for reading in particle data (fort.14) from upstream
domain (this allows transferring the outlet particle data
from the upstream domain solutions to the inlet
boundary for succeeding domain computations --
especially useful for multi-phase rocket plume
simulations)

<table>
<thead>
<tr>
<th>Card Group #19</th>
<th>Particle initial conditions (for steady-state runs only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>(2 lines*IDPTCL)</td>
</tr>
<tr>
<td></td>
<td>IPTZON, IDBCPT, LPTCL1, LPTCL2, MPTCL1, MPTCL2, NPTCL1, NPTCL2, ITPTCL, DDPTCL, DNPTCL, WDMASS, UUPTCL, HTPTCL</td>
</tr>
<tr>
<td>Definition</td>
<td>IPTZON</td>
</tr>
<tr>
<td></td>
<td>IDBCPT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPTCL1, LPTCL2</td>
</tr>
<tr>
<td></td>
<td>MPTCL1, MPTCL2</td>
</tr>
<tr>
<td></td>
<td>NPTCL1, NPTCL2</td>
</tr>
</tbody>
</table>
|                | ITPTCL         | number of particle groups (trajectories) starting
|                |                | from each grid cell |
|                | DDPTCL         | particle diameter in μm |
|                | DNPTCL         | particle density in lbm/ft³ |
|                | WDMASS         | particle mass flow rates for the current particle
group and area involve of the current input line |
|                | UUPTCL         | particle/gas velocity ratio at the initial positions |
|                | HTPTCL         | particle initial enthalpy in ft²/sec² |
There are two entries in the input file: 1) FLUID, and 2) PARALLEL. The “FLUID” entry is needed when the homogeneous real-fluid model is activated (INSO(7) > 0), and the “PARALLEL” entry is needed if the parallel computing will be conducted. These two entries have to be located after all card groups. The sample format for these two entries is also included in Appendix A. In the “FLUID” entry, there are parameters to control the debugging outputs and the number of sub-iterations, as well as to identify the species which properties will be calculated by the real-fluid model. In addition, a database file, “dbase.dat”, which contains thermodynamic properties for all chemical species needs to reside in the same working directory as input data files. In the “PARALLEL” entry, there are three input lines for the processor(s) specified in the Card Group #3. For the first input line, it starts with the index number(s) of the processor(s), such as 2 or 2-3 and etc., and followed by “host=.....” to identify the hostname of the machine where the processor(s) reside. The second input line, “edir=.....” is to identify the path to the directory (or file folder) which contains the executable of the FDNS-RFV/PVM code. Meanwhile, the path to the directory (or file folder) which contains all the input and output files will be defined as “wdir=.....” in the third input line. It is to be noted that 1) the master host must be the first host machine (i.e. in the first line), 2) each line in the “PARALLEL” entry can not exceed 160 columns, 3) the “PARALLEL” entry starts with a line contains “parallel” and must end with a line which has “end of parallel”, and 4) all input data files (including fort.11, fort.12, fort.13, and dbase.dat) have to reside in the working directory (designated by “wdir=.....”) of the master host.

### 3.3 PRE-/POST-PROCESSOR FOR PARALLEL COMPUTING

The first step of conducting the parallel computing is to activate the PVM daemon on the master and slave machines. This can be done either by typing “pvm” on the master machine and then adding hostnames of the slave machine which the employed processor is located on, or by typing “pvm hostfile” on the master machine, where the file—“hostfile” contains the hostnames of the slave machine which the employed processor is located on. Before running the FDNS-RFV/PVM code with the parallel computing, users have to run the pre-processor, “xprep”, on the master machine to decompose each input data file into several sub-domain input data files for each employed processor, and then distribute them to the corresponding master host or slave host where the processor is located. The sub-domain for each processor is prescribed in “fort.11”. In order to perform the domain decomposition, the master pre-processor, “xprep”, have to reside in the master host, while the slave pre-processor, “fpvm”, have to be located in the directory (designated by “edir=.....”) of each slave host. For example, the domain decomposition program will read in input data file—“fort.11” and then produce a sub-domain input data files (such as f001.11, f002.11, f003.11, and etc.) on the master where the corresponding processors (such as processor #1, #2, #3, and etc.) are located. The decomposed sub-domain input data files of the sample input data file, as shown in Appendix A, are listed in Appendix B for illustration. Same logic is applied to other input data files, such as fort.12, fort.13, and etc. The combination of “xprep” and “fpvm” can also serve as the post-processor, which reverses the previous procedure to read in the output data files (such as f001.23, f002.23, and etc.) of each processor and generate the output data files of the whole domain. To run the FDNS-RFV/PVM, users can simply type the name of the compiled executable (e.g. “PATH/xfdns”) in the working directory, where all the input data files reside in, followed by either “-p” option for parallel computing or “-np” option for non-parallel computing. “PATH” can be either the absolute path name from users’ home directory to the directory contains the executable, or the relative path name from the working directory to the directory where the executable resides in.
3.4 USER DEFINED RUN-TIME MODIFICATIONS

Three run-time modification include file are used for the current version of the FDNS-RFV/PVM code. They are: fmain01, fmain02 in the main program and fexmp01 in the subroutine EXAMP. All these include files are entered in the fdns.f file. Only this file needs to be re-compiled after any of these three include files are changed. However, if any of the COMMON block include files, fdns01 through fdns17, and fluid.inc, is changed, the entire code must be re-compiled. Samples of the usage of these include files for a given example problem are listed in Appendix C.

3.4.1 Main Program Includes (fmain01 & fmain02)

The include file fmain01 is only used after the restart files are read. Any modifications to the restarted data such as the inlet pressure level setting, inlet velocity profile modification, re-initializing part of the flowfield and/or wall temperature resetting, etc. FORTRAN statement numbers between 7000 and 7900 can be used in the coding of this include file.

The second include file fmain02 is entered after every time step. Any run-time modification such as boundary condition adjustment, grid modification (however, DIRCOS and TRANF must be called after grid modification), and run-time print of any data of interest (file unit numbers between 30 and 89 are recommended for printing out the user's data). FORTRAN statement numbers between 8000 and 8900 can be used in the coding of fmain02.

3.4.2 Example Subroutine Includes (fexmp01)

In the example start include file, fexmp01 (which is included in the subroutine EXAMP), one can include simple grid generation and initial flowfield specification FORTRAN coding to start a problem. Subroutines IVA4, RVA4 are used to simplify the coding and subroutine XISENT can be employed to generate initial nozzle flowfield based on 1-D isentropic relations. Another way of using the fexmp01 file, such as the one used in the Example Problems section, is to write a simple but general grid and flow data read-in code in the fexmp01 file. Then a separate grid generation and flowfield initialization program is written for generating the grid and flow data file which is then used by the fexmp01 data input code.

3.5 Restart/Output Files (in Main, DATINN, and DATOUT)

The grid and flowfield restart input files (fort.12 & fort.13) and output files (fort.22 & fort.23) are handled in the subroutines DATINN and DATOUT. The subroutine DATP3D prints out one PLOT3D grid-file (fort.91) and two PLOT3D q-files (through fort.92 and fort.93). The PLOT3D grid data are not re-scaled, except for 3-D pump problems where grid data are normalized by the pump tip diameter. The first PLOT3D q-file, fort.92, includes the following 5 variables:

- \( q_1 \) Density in \( \text{lb m/ft}^3 \) or unity
- \( q_2 \) U-Velocity in \( \text{ft/sec} \) or normalized by the pump tip speed
- \( q_3 \) V-Velocity in \( \text{ft/sec} \) or normalized by the pump tip speed
- \( q_4 \) W-Velocity in \( \text{ft/sec} \) or normalized by the pump tip speed
q5  Non-dimensional total pressure.

The second PLOT3D q-file, fort.93, includes the following 5 variables:

q1  Density in lbm/ft³ or unity
q2  Pressure in psia
q3  Temperature in EK
q4  Mach number
q5  First species mass-fraction, or quality of the specified species

Units of the PLOT3D outputs can be modified by editing the DATAIO subroutine. Unformatted data are used in DATAIO subroutine. Other formats can be incorporated by modifying DATAIO. The TECPLOT data file contains both grid coordinates and flow variables, which are listed as: X, Y, Z (3-D only), U, V, W (3-D only), pressure, density, temperature, Mach number, quality (if real-fluid model is activated), and species mass fractions. The units of all variables are determined through the input card (fort.11).
A sample input data file fort.11 for the example problem of a vortex engine simulation is shown in the following. The vortex engine employs 16 pairs of unlike doublets, and uses LOX and RP-1 as the propellants which are injected tangentially along the chamber side wall. The propellant injecting sequence of the unlike doublet is alternating in the configuration of this sample test case. The grid and initial flowfield generator is illustrated in Appendix C.

| TITLE: (Vortex Chamber test case #2) |
| IDIM, 3, |
| IZON, IZFACE, IBND, ID, TSNGL, 2, 3, 9, 4, 2, |
| IZT, JZT, KZT, LPROC, CBGX, CBGY, CBGZ, CBVX, CBVY, CBVZ, 41, 41, 61, 1, 0.0, 0.0, 0., 0., 0., 0., |
| 67, 61, 61, 2, 0.0, 0.0, 0., 0., 0., 0., 0., |
| NNBC, IZB1, IZP1, IJZ1, IJZ2, JZK1, JZK2, INONUF, (2 LINES EACH) |
| IZB2, IZP2, IJZ1, IJZ2, JZK1, JZK2, |
| 1, 1, 1, 1, 26, 1, 61, 0, |
| 2, 2, 1, 26, 1, 61, |
| -1, 1, 5, 1, 41, 1, 41, 0, |
| 1, 6, 1, 41, 1, 41, |
| -1, 2, 5, 1, 67, 1, 26, 0, |
| 2, 6, 1, 67, 1, 26, |
| IBCZON, IDBC, ITYBC, IJBB, IJBS, IJBT, IKBS, IKBT, 1, 3, -1, 41, 13, 16, 13, 31, |
| 1, 3, -1, 41, 28, 31, 7, 31, |
| 1, 3, -1, 41, 26, 11, 14, 37, 61, |
| 1, 3, -1, 41, 26, 29, 43, 61, |
| 1, 3, -1, 41, 26, 29, 1, 1, |
| 1, 4, 3, 1, 1, 41, 1, 61, |
| 2, 4, 3, 1, 1, 67, 1, 61, |
| 2, 1, 2, 67, 1, 26, 1, 61, |
| IWZON, L1, L2, M1, M2, N1, N2, IWTM, HQDOX, IWALL, DENNX, VISWX, 1, 1, 1, 1, 41, |
| 1, 1, 1, 41, 1, 61, 1, 0., 0, 1., 1., |
| 1, 1, 41, 41, 41, 1, 61, 1, 0., 0, 1., 1., |
| 1, 41, 41, 26, 41, 1, 61, 1, 0., 0, 1., 1., |
| 2, 1, 67, 26, 26, 1, 61, 1, 0., 0, 1., 1., |
| ISNZN, ISNBC, ISNAX, ISNBS, ISNBT, 1, 4, 1, 1, 41, |
| 2, 4, 1, 1, 67, |
| IDATA, IGEO, ITT, ITPNT, ICOUP, NLIMT, IAX, ICYC, 1, 9, 10000, 20, 1, 1, 1, 3, |
| DTT, IREC, REC, THETA, BETAP, IEXX, PRAT, 1.000E-03, 0, 0.25, 1.0, 1.00, 1, -1.0, |
| IPC, JPEX, IMN, JMN, 31928, 1, 3417, 2, 50329, 1, |
| VIS(1/RE), IG, ITURB, AMC, GAMA, CBE, CBH, EREXT, 2.84554E-06, 2, 1, 0.040, 1.455, 0.0, 0.0, 1.E-09, |
| ISWU, ISWP, ISWK, ISKEW, 93, 97, 93, 0, |
| INSO (IEQ): (VISCOSITY = 4.4228260E-07 SLUGS/FT-SEC) |
| U, V, W, TM, DK, DE, 7, 8, 9, VS, FM, SP, 1, 1, 1, 1, 1, 1, 1, 10, 0, 0, 1, 1, 32, |
| ---NGAS, NREACT, IUNIT, DREF(SLG), UREF(F/S), TREF(R), XREF (FT), PREF |
11, 15, 2, 2.2708E-03, 228.3236, 540.00, 0.0833333, 1000.
IGDINN, IOFINN, IOFOUT, IOP3DOUT (1:Unf, 2:Fmt) ! IO FORMAT CONTROL
1, 1, 1, 0
H2O
0.26340654E+01 0.31121899E-02-0.00278451E-06 0.12673054E-09-0.69164734E-14
-0.29876258E+05 0.70823874E+01 0.41675563E+01-0.00181068E-06 0.59450877E-05
-0.48670872E-08 0.15284144E-11-0.30289547E+05-0.73087996E+00
O2
0.36122139E+01 0.74853166E-03-0.19820646E-06 0.33759616E-02-0.99427545E-05
-0.98199101E-08 0.30318266E-11-0.10638107E+05-0.73087996E+00
CO
0.29840696E+01 0.14891390E-02-0.57899683E-06 0.10364577E-09-0.69353550E-14
-0.14245228E+05 0.63479156E+01 0.37100928E+01-0.16190964E-02 0.36923593E-05
-0.20319675E-08 0.23953344E-12-0.14356310E+05 0.29555352E+01
CO2
0.44608040E+01 0.74853166E-03-0.19820646E-06 0.33759616E-02-0.99427545E-05
-0.98199101E-08 0.30318266E-11-0.10638107E+05-0.73087996E+00
H2
0.30558124E+01 0.59740403E-03-0.16747471E-08-0.21247544E-10 0.25195486E-14
-0.86168475E+03-0.17207073E+01 0.29432328E+01 0.34815508E-02-0.77713821E-05
-0.74997493E-08-0.25203379E-11-0.97695410E+03-0.18186136E+01
O
0.25342960E+01 0.10152035E-02-0.22048808E-06 0.20195486E-10-0.39409830E-15
0.38632414E+04 0.55566425E+01 0.41826975E+13-0.13397373E-02 0.16348351E-05
-0.52133636E-09 0.41826975E-13 0.35802349E+04 0.34204206E+00
OH
0.25474391E+05-0.45989841E+00 0.25000000E+01 0.00000000E+00 0.00000000E+00
-0.10187399E-07 0.43603341E-11-0.78824560E+04 0.15699718E+02
C2H4
0.43843679E+01 0.96509494E-02-0.31845530E-05 0.46213655E-09-0.24069342E-13
-0.91048380E+04-0.23853559E+01 0.11654673E+01 0.13374992E-01 0.25754771E-05
-0.75958920E-07 0.27631500E+03-0.33154709E+02 0.61848474E+00-0.37158785E-04
SOOT
0.24214220E+02 0.76643730E-01-0.30868710E+04 0.57363350E-08-0.39918970E-12
-0.56446757E+05-0.28713500E+03 0.40000000E+01 0.88877200E-01 0.31025740E-04
-0.75958920E-07 0.27636150E+10-0.48169200E+05-0.17046840E+03
SOOT
0.24214220E+02 0.76643730E-01-0.30868710E+04 0.57363350E-08-0.39918970E-12
-0.56446757E+05-0.28713500E+03 0.40000000E+01 0.88877200E-01 0.31025740E-04
-0.75958920E-07 0.27636150E+10-0.48169200E+05-0.17046840E+03
SOOT
0.24214220E+02 0.76643730E-01-0.30868710E+04 0.57363350E-08-0.39918970E-12
-0.56446757E+05-0.28713500E+03 0.40000000E+01 0.88877200E-01 0.31025740E-04
-0.75958920E-07 0.27636150E+10-0.48169200E+05-0.17046840E+03
REACTION:H2O, O2, CO, H2, O, H, OH, C2H4, RP-1, SOOT
1, 3.0117E10, 0.00, 2.5232E4, 0, 2
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 6.2, -1.0, 0.0, 0.0, 0.0, -0.5, 0.0
2, 1.2900E15, -1.00, 2.5161E4, 0, 2
0.0, -1.0, 2.0, 0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0, -0.5, 0.0
3, 5.1308E12, 2.00, 1.6114E4, 0, 2
0.0, 0.0, 0.0, 84.0, 0.0, 0.0, -0.48, 0.0, 0.0, 0.0, -2.00, 0.0, 0.0
2, 1.7000E13, 0.00, 24070.0, 0, 0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
FDNS-RFV/PVM

0., -1., 0., 0., -1., 0., 0., 2., 0., 0., 0., 0.
3, 2.1900E13, 0.00, 2590., 0, 0,
  1., 0., 0., 0., 0., -1., 0., 1., -1., 0., 0., 0.
4, 6.0230E12, 0.00, 550., 0, 0,
  1., 0., 0., 0., 0., 0., 1., 0., -2., 0., 0., 0.
5, 1.8000E10, -1.00, 4480., 0, 0,
  1., 0., 0., 0., -1., -1., 1., 1., 0., 0., 0., 0.
6, 1.2200E17, 0.91, 8369., 0, 0,
  0., -1., 0., 0., 0., 1., -1., 1., 0., 0., 0., 0.
7, 4.0000E12, 0.00, 4030., 0, 0,
  0., 0., -1., 1., 0., 0., 0., 1., -1., 0., 0., 0.
8, 3.0000E12, 0.00, 25000., 0, 0,
  0., -1., -1., 1., 0., 1., 0., 0., 0., 0., 0., 0.
9, 1.0000E16, 0.00, 9999., 0,
  0., 0., 0., 0., 0., -1., -1., 1., 0., 0., 0., 0.
10, 2.5500E18, 1.00, 59390., 999, 0,
    0., 1., 0., 0., 0., -2., 0., 0., 0., 0., 0., 0.
11, 5.0000E15, 0.00, 9999., 0,
    0., 0., 0., 0., 1., 0., -2., 0., 0., 0., 0., 0.
12, 8.4000E21, 2.00, 0.999, 0,
    0., 0., 0., 0., 0., 0., 0., -1., -1., 0., 0., 0.
13, 6.0000E13, 0.00, 0.999, 0,
    0., -1., -1., 1., 0., -1., 0., 0., 0., 0., 0., 0.

***END******
FLUID
c Species(a20) , ideal gas(=0), real fluid(=1)
H2O        0
O2         1
CO         0
CO2        0
H2         0
O          0
H          0
OH         0
C2H4       0
RP-1       1
SOOT       0
DONE       0
PARALLEL
1 host=gemini.seca-hsv.com
edir=/home/gary/fdnsrc-f-pvm
  wdir=/home/gary/fdnsrc-f-pvm/vortex2
2 host=pisces.seca-hsv.com
  edir=/home/gary/fdnsrc-f-pvm
  wdir=/home/gary/fdnsrc-f-pvm/vortex2
end of parallel

57
APPENDIX B

The decomposed sub-domain input data files for each processor, generated from the preprocessor program for the parallel computing, “xprep” & “fpvm”, with the sample input data file fort.11 listed in Appendix A, are shown in the following.

<< Input File for Processor #1: f001.11 >>

```
TITLE: : (Vortex Chamber test case #2)
IDIM
3
IZON,IZFACE, IBND, ID, ISNGL
 1  2  7  3  1
I2T, J2T, K2T,LPROC, CBGX, CBGY, CBGZ, CBVX, CBVY, CBVZ
 41  41  61  1  0.0E+00  0.0E+00  0.0E+00  0.0E+00  0.0E+00  0.0E+00
IFCYC, IZB1, IZF1, IJZ11, IJZ12, JKZ11, JKZ12, INONUF, IPROC1
 IZB2, IZF2, IJZ21, IJZ22, JKZ21, JKZ22, IDFACE, IPROC2
  1  1  1  26  1  61  51  1
  1  2  1  26  1  61  1  2
  1  5  1  41  1  41  0  1
  6  1  41  1  41  2  1
IZCZON, IDBC, ITYBC, IJBB, IJBS, IJBT, JKBS, JKBT, IBG
 1  3  -1  41  13  16  13  31  1
 1  3  -1  41  28  31  7  31  2
 1  3  -1  41  11  14  37  61  3
 1  3  -1  41  26  29  43  61  4
 1  3  -1  41  11  14  1  1  5
 1  3  -1  41  26  29  1  1  6
 1  4  3  1  1  41  1  61  7
IWZON, L1, L2, M1, M2, N1, N2,IWTM, HQDOX,IWALL, DENNX, VISWX
 1  1  1  41  1  61  1  0.000E+00  0.000E+00  1.000E+00  1.000E+00
 1  41  41  1  61  1  0.000E+00  0.000E+00  1.000E+00  1.000E+00
 1  41  41  26  41  1  61  1  0.000E+00  0.000E+00  1.000E+00  1.000E+00
ISZON, ISNBC, ISNAX, ISNBS, ISNBT
 4  1  1  41
IDATA, IGEO, ITT, ITPNT, ICOUP, NLIMT, IAX, ICYC
 1  9 10000  20  1  1  1  1  3
DTT, IREC, REC, THETA, BETAP, IEXX, PRAT
 1.000E-03  0  .2500  1.0000  1.0000  1.0000  1.0000  1.0000  1.0000  1.0000
IPC, JPC, IPEX, JPEX, INN, JMN
 31928  1  3417  -2  50329  1
VISC, IG, ITURB, AMC, GAMG, CBE, CBH, EREXT
 2.84554E-06  2  1  .0400  1.4550  0.000E+00  0.000E+00  1.000E-09
ISWU, ISWP, ISWK, ISKEW
 93  97  93  0
INSO(IEQ):
U, V, W, T, DK, DE, 07, 08, 09, VS, FM, SP
 1  1  1  1  1  1  1  10  0  0  1  1  32
NGAS, NREACT, IUNIT, DENREF, UREF, TREF, XREF, PREF,
11  15  2  2.271E-03  2.283E+02  5.400E+02  8.333E-02  1.000E+03
IGDINN, IOFINN, IOFOUT, IOP3DOUT (1:Unf, 2:Fmt) ! IO FORMAT CONTROL
 1  1  1  0
H2O                                                                   18.01520
2.63406539E+00  3.11218994E-03-9.02784507E-07 1.26730543E-10-6.91647337E-15

58
```
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Rate Constant</th>
<th>Stoichiometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2O + O2 + CO + CO2 + H2 + O + H + OH + C2H4 + RP-1 + SOOT</td>
<td>3.0117E+10</td>
<td>0.0000E+00 2.5230E+04 0 2</td>
</tr>
<tr>
<td>O2 + CO + CO2 + H2 + O + H + OH + C2H4 + RP-1 + SOOT</td>
<td>1.2900E+15</td>
<td>-1.0000E+00 2.5160E+04 0 2</td>
</tr>
<tr>
<td>O2 + CO + CO2 + H2 + O + H + OH + C2H4 + RP-1 + SOOT</td>
<td>5.1308E+12</td>
<td>2.0000E+00 1.6110E+04 0 2</td>
</tr>
<tr>
<td>O2 + CO + CO2 + H2 + O + H + OH + C2H4 + RP-1 + SOOT</td>
<td>1.7000E+13</td>
<td>0.0000E+00 2.4070E+04 0 2</td>
</tr>
<tr>
<td>O2 + CO + CO2 + H2 + O + H + OH + C2H4 + RP-1 + SOOT</td>
<td>2.1900E+13</td>
<td>0.0000E+00 2.5900E+03 0 2</td>
</tr>
<tr>
<td>O2 + CO + CO2 + H2 + O + H + OH + C2H4 + RP-1 + SOOT</td>
<td>6.0230E+12</td>
<td>0.0000E+00 5.5000E+02 0 2</td>
</tr>
<tr>
<td></td>
<td>7E+10</td>
<td>-1.0000E+00</td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>1.8E+10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.2E+17</td>
<td>9.1000E-01</td>
</tr>
<tr>
<td></td>
<td>1.2E+16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4.0E+12</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td></td>
<td>1.8E+16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3.0E+12</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td></td>
<td>1.8E+16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.0E+16</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td></td>
<td>2.5E+18</td>
<td>1.0000E+00</td>
</tr>
<tr>
<td></td>
<td>5.0E+15</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td></td>
<td>8.4E+21</td>
<td>2.0000E+00</td>
</tr>
<tr>
<td></td>
<td>6.0E+13</td>
<td>0.0000E+00</td>
</tr>
</tbody>
</table>

**FLUID**

Species(a20), ideal gas (=0), real fluid (=1)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H2O</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>O2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CO</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CO2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>H2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OH</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C2H4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RP-1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SOOT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DONE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<< Input File for Processor #2: f002.11 >>

TITLE: : (Vortex Chamber test case #2)
IDIM
3
IZON,IZFACE, IBND, ID, ISNGL
1 2 2 1 1
IZT, JZT, KZT, LPROC, CBGX, CBGY, CBGZ, CBVX, CBVY, CBVZ
67 26 61 2 0.0E+00 0.0E+00 0.0E+00 0.0E+00 0.0E+00 0.0E+00
IFCYC, IZB1, IZF1, IZJ21, IJZ21, JZK21, JKZ22, IDFACE, IPROC1
1 1 1 1 26 1 61 51 1
1 2 1 26 1 61 1 2
-1 1 5 1 67 1 26 0 2
1 6 1 67 1 26 3 2
IZB2, IZF2, IZJ22, JKZ22, IDFACE, IPROC2
1 1 2 1 26 1 61 51 1
1 2 1 26 1 61 1 2
IBCZON, IDB, ITYBC, IJBB, IJBS, JKBS, JKB, IBG
1 4 3 1 1 67 1 61 8
1 1 2 67 1 26 1 61 9
IWBZON, LI, L2, M1, M2, N1, N2, IWTM, HQDOX, IWTM, DENNX, VISWX
1 1 67 26 26 1 61 1 0.000E+00 0 1.000E+00 1.000E+00
ISNZON, ISNBC, ISNAX, ISNBS, ISNBT
1 4 1 1 67
IDATA, IGEO, ITT, IPTNT, ICOUP, NLIMT, IAX, ICYC
1 9 10000 20 1 1 1 3
DTT, IREC, REC, THETA, BETAP, IEXX, PRAT
1.000E-03 0 .2500 1.0000 1.0000 1 -1.00
IPC, JPC, IPEX, JPEX, IMN, JMN
31928 -1 3417 1 50329 -1
VISC, IG, ITURB, AMC, GAMMA, CBE, CBH, EREXT
2.84554E-06 2 1 .0400 1.4550 0.000E+00 0.000E+00 1.000E-09
ISWU, ISWP, ISWK, ISKEW
93 97 93 0
INSO(IEQ):
U, V, W, T, DK, DE, 07, 08, 09, VS, FM, SP
1 1 1 1 1 1 10 0 0 1 1 32
NGAS, NREACT, IUNIT, DENREF, UREF, TREF, XREF, PREF,
11 15 2 2.271E-03 2.283E+02 5.400E+02 8.333E-02 1.000E+03
IGDINN, IOFINN, IOFOUT, IOP3DOUT (1:Unf, 2:Fmt) ! IO FORMAT CONTROL
1 1 0
H2O
2.63406539E+00 3.11218994E-03-9.02784507E-07 1.26730543E-10-6.91647337E-15
-2.98762578E+04 7.08238745E+00 4.16755629E+00-1.81068678E-03 5.94508765E-06
-4.86708718E-09 1.52841444E-12-3.02895469E-07 1.03681067E+03 3.64163446E+00
O2
3.61221385E+00 7.48531660E-04-1.98206465E-07 3.37490071E-11-2.39073736E-15
-1.19781506E+03 3.67033076E+00 3.78371358E+00-3.02336342E-03 3.94927541E-06
-9.81891013E-09 3.03318260E-12-1.06381067E+03 3.64163446E+00
H2
2.01580

61
FDNS-RFV/PVM

![Image of the text content]
1.0  0.0  0.0  0.0  -1.0 -1.0  0.0  0.0  0.0
15  6.0000E+13  0.0000E+00  0.0000E+00  999  0
   0.0 -1.0  1.0  0.0 -1.0  0.0  0.0  0.0  0.0

FLUID

c     Species(a20)  , ideal gas(=0), real fluid(=1)
H2O   0
O2    1
CO    0
CO2   0
H2    0
O     0
H     0
OH    0
C2H4  0
RP-1  1
SOOT  0
DONE
APPENDIX C

There are three modules in this Appendix. The first module, "fexmp01", is a sample program to read in the grid and initial flowfield for the example problem of a vortex engine simulation. The second module, "fmain02", is a sample program to print out some key flow variables for the purpose of monitoring convergence history and post-processing. The third module is a sample program to generate the grid and initial flowfield for the example problem.

1. List of fexmp01

C-----example file for the vortex engine
IF(IGEO .EQ. 9) THEN

C------- READ IN PRE-PROCESSED GRIDS AND INITIAL FLOW FIELD
IR4 = 14
IR5 = 15
READ(IR4,1) IZN
IZS(1)=0
DO 4310 IZ=1,IZN
   READ(IR4,1) IZT(IZ),JZT(IZ),KZT(IZ)
   IF(IZ.GE. 2)IZS(IZ)=IZS(IZ-1)+IZT(IZ-1)*JZT(IZ-1)*KZT(IZ-1)
4310  CONTINUE
DO 4320 IZ=1,IZN
   L=IZS(IZ)
   READ(IR4,2)(X(L+IJK),IJK=1,LMN)
   READ(IR4,2)(Y(L+IJK),IJK=1,LMN)
   READ(IR4,2)(Z(L+IJK),IJK=1,LMN)
4320  CONTINUE

C
READ(IR5,3) JUKN1,JUNK2,JUNK3,JUNK4,JUNK5
DO 4330 IZ=1,IZN
   L=IZS(IZ)
   READ(IR5,2)(DEN(L+IJK),IJK=1,LMN)
   READ(IR5,2)(U(L+IJK),IJK=1,LMN)
   READ(IR5,2)(V(L+IJK),IJK=1,LMN)
   READ(IR5,2)(W(L+IJK),IJK=1,LMN)
   READ(IR5,2)(P(L+IJK),IJK=1,LMN)
   READ(IR5,2)(TM(L+IJK),IJK=1,LMN)
   READ(IR5,2)(DK(L+IJK),IJK=1,LMN)
   READ(IR5,2)(DE(L+IJK),IJK=1,LMN)
   IF(NGAS .GT. 0) THEN
      DO 4340 ISP=1,NGAS
         READ(IR5,2)(FM(L+IJK,ISP),IJK=1,LMN)
      4340    CONTINUE
      DO 4350 IJK=1,LMN
         FMSUM=0.
         DO 4360 ISP=1,NGAS
            FMSUM=FM(IJK,ISP)+FMSUM
         4360      CONTINUE
         IF(ABS(FMSUM-1.) .GT. 1.E-4) THEN
WRITE(6,4301) IJK,FMSUM
STOP
END IF
END IF
CONTINUE
C
4301 FORMAT('@@@ MASS FRACTION ERROR AT IJK =',I6,', FM = ',F10.6)
ENDIF
C
1 FORMAT(8I5)
2 FORMAT(5(1P,E16.8))
3 FORMAT(15I5)

2. List of fmain02

C***** PRINT OUT PROFILE DATA **************C
C----- (USE STATEMENT NUMBER 8000 TO 8900)
C
C-------- Vortex Engine test case #2
C---- Reset the Wall corner Points
IF(NPROC .LE. 1 .OR. IPROC .LE. 1) THEN
  IZ=1
  L=IZS(IZ)
  M=JZS(IZ)
  N=KZS(IZ)
  J=JZT(IZ)
  I=1
DO K=1,KZT(IZ)
  IJK  = L+I+(J-1)*M+(K-1)*N
  IJK1 = IJK+1
  IJK2 = IJK-M
  TM(IJK)=0.5*(TM(IJK1)+TM(IJK2))
  P(IJK)=0.5*(P(IJK1)+P(IJK2))
  DEN(IJK)=0.5*(DEN(IJK1)+DEN(IJK2))
  FMDUM=0.
  DO KK=1,NGAS
    FM(IJK,KK)=0.5*(FM(IJK1,KK)+FM(IJK2,KK))
    FMDUM=FMDUM+FM(IJK,KK)
  ENDDO
  DO KK=1,NGAS
    FM(IJK,KK)=FM(IJK,KK)/FMDUM
  ENDDO
ENDDO
ENDDO
I=IZT(IZ)
DO K=1,KZT(IZ)
  IJK  = L+I+(J-1)*M+(K-1)*N
  IJK1 = IJK-1
  IJK2 = IJK-M
  TM(IJK)=0.5*(TM(IJK1)+TM(IJK2))
  P(IJK)=0.5*(P(IJK1)+P(IJK2))
  DEN(IJK)=0.5*(DEN(IJK1)+DEN(IJK2))
  FMDUM=0.
  DO KK=1,NGAS
    FM(IJK,KK)=FM(IJK,KK)/FMDUM
  ENDDO
ENDDO
ENDDO
I=IZT(IZ)
DO K=1,KZT(IZ)
  IJK  = L+I+(J-1)*M+(K-1)*N
  IJK1 = IJK-1
  IJK2 = IJK-M
  TM(IJK)=0.5*(TM(IJK1)+TM(IJK2))
  P(IJK)=0.5*(P(IJK1)+P(IJK2))
  DEN(IJK)=0.5*(DEN(IJK1)+DEN(IJK2))
  FMDUM=0.
  DO KK=1,NGAS

FM(IJK, KK) = 0.5 * (FM(IJK1, KK) + FM(IJK2, KK))
FMDUM = FMDUM + FM(IJK, KK)
ENDDO
DO KK = 1, NGAS
  FM(IJK, KK) = FM(IJK, KK) / FMDUM
ENDDO
ENDDO
ENDDO
ENDIF
C
IF(MOD(ITO, ITPNT) .EQ. 0) THEN
  TMMAX = 0.
  TMMIN = 1.E30
  FMMAX = 0.
  IJKFMX = 1
  DO IJK = 1, IGDMAX
    IF(TM(IJK) .GT. TMMAX) THEN
      TMMAX = TM(IJK)
      HTTMX = HT(IJK)
      PRTMX = P(IJK)
      DO KK = 1, NGAS
        FMTMX(KK) = FM(IJK, KK)
      ENDDO
      IJKTMX = IJK
    END IF
    IF(TM(IJK) .LT. TMMIN) THEN
      TMMIN = TM(IJK)
      HTTMN = HT(IJK)
      PRTMN = P(IJK)
      DO KK = 1, NGAS
        FMTMN(KK) = FM(IJK, KK)
      ENDDO
      IJKTMN = IJK
    END IF
    IF(FM(IJK, 11) .GT. FMMAX) THEN
      FMMAX = FM(IJK, 11)
      TMFMX = TM(IJK)
      HTFMX = HT(IJK)
      PRFMX = P(IJK)
      DO KK = 1, NGAS
        FMFMX(KK) = FM(IJK, KK)
      ENDDO
      IJKFMX = IJK
    END IF
  ENDDO
MTAG = 88000
IFMTMN = 1
IFMTX = 1
IFPFMX = 1
CALL AINDEX(IJKTMN, ITZN, ITMN, JTMN, KTMN)
CALL AINDEX(IJKTMX, ITZN, JTMX, KTMX)
CALL AINDEX(IJKFMX, ITZN, JFMX, KFMX)
IF(IPROC .LE. 1) THEN
  DO IP = 2, NPROC
    CALL RECVIT4(ITMN2, JTMN2, KTMN2, ITZN2, ITIDS(IP), MTAG)
    CALL RECVIT4(ITMX2, JTMX2, KTMX2, ITZN2, ITIDS(IP), MTAG)
    CALL RECVIT4(IFMX2, JFMX2, KFMX2, ITZN2, ITIDS(IP), MTAG)
    CALL RECEBN3(TMMN2, HTTMN2, PRTMN2, ITIDS(IP), MTAG)
    CALL RECEBN3(TMMX2, HTTMX2, PRTMX2, ITIDS(IP), MTAG)
  ENDDO
CALL RECVNB4 (FMMAX2, TMFMX2, HTFMX2, PRFMX2, ITIDS(IP), MTAG)
CALL RECVNB (FMTMN2, 1, NSPM, ITIDS(IP), MTAG)
CALL RECVNB (FMTMX2, 1, NSPM, ITIDS(IP), MTAG)
CALL RECVNB (FMFMX2, 1, NSPM, ITIDS(IP), MTAG)
IF (TMMIN2 .LT. TMMIN) THEN
  IPTMN = IP
  TMMIN = TMMIN2
  HTTMN = HTTMN2
  PRTMN = PRTMN2
  DO KK=1,NGAS
    FMTMN(KK) = FMTMN2(KK)
  ENDDO
  ENDDO
  CALL IVA4 (ITMN, JTMN, KTMN, IZTMN, ITMN2, JTMN2, KTMN2, IZTMN2)
ENDIF
IF (TMMAX2 .GT. TMMAX) THEN
  IPTMX = IP
  TMMAX = TMMAX2
  HTTMX = HTTMX2
  PRTMX = PRTMX2
  DO KK=1,NGAS
    FMTMX(KK) = FMTMX2(KK)
  ENDDO
  ENDDO
  CALL IVA4 (ITMX, JTMX, KTMX, IZTMX, ITMX2, JTMX2, KTMX2, IZTMX2)
ENDIF
IF (FMMAX2 .GT. FMMAX) THEN
  IPFMX = IP
  FMMAX = FMMAX2
  TMFMX = TMFMX2
  HTFMX = HTFMX2
  PRFMX = PRFMX2
  DO KK=1,NGAS
    FMFMX(KK) = FMFMX2(KK)
  ENDDO
  ENDDO
  CALL IVA4 (IFMX, JFMX, KFMX, IZFMX, IFMX2, JFMX2, KFMX2, IZFMX2)
ENDIF
ENDIF
TMMIN = TMMIN * TREF1
WRITE (6, 8440) IPTMN, IZTMN, ITMN, JTMN, KTMN, TMMIN, HTTMN,
  & PRTMN * QREF1 / 144., (FMTMN(KK), KK=1, NGAS)
TMMAX = TMMAX * TREF1
WRITE (6, 8440) IPTMX, IZTMX, ITMX, JTMX, KTMX, TMMAX, HTTMX,
  & PRTMX * QREF1 / 144., (FMTMX(KK), KK=1, NGAS)
IF (NREACT .GT. 0) THEN
  TMPFMX = TMFMX * TREF1
  FMO = FMFMX(1) * 8./9. + FMFMX(2) + FMFMX(3) * 4./7.
  & + FMFMX(4) * 8./11. + FMFMX(6) * 16./17.
  & + FMFMX(3) * 3./7. + FMFMX(4) * 3./11.
  WRITE (6, *) '** AT MAX Soot POINT, FMC & FMO = ', FMC, FMO
WRITE (6, 8440) IPFMX, IZFMX, IFMX, JFMX, KFMX, IFMX2, JFMX2, KFMX2, IZFMX2
ENDIF
ELSE
  CALL SENDIT4 (ITMN, JTMN, KTMN, IZTMN, ITIDS(1), MTAG)
  CALL SENDIT4 (ITMX, JTMX, KTMX, IZTMX, ITIDS(1), MTAG)
  CALL SENDIT4 (IFMX, JFMX, KFMX, IZFMX, ITIDS(1), MTAG)
  CALL SENDNB3 (TMMIN, HTTMN, PRTMN, ITIDS(1), MTAG)
  CALL SENDNB3 (TMMAX, HTTMX, PRTMX, ITIDS(1), MTAG)
  CALL SENDNB4 (FMMAX, TMFMX, HTFMX, PRFMX, ITIDS(1), MTAG)
ENDIF
CALL SENDNB (FMTMN, 1, NSPM, ITIDS(1), MTAG)
CALL SENDNB (FMTMX, 1, NSPM, ITIDS(1), MTAG)
CALL SENDNB (FMFMX, 1, NSPM, ITIDS(1), MTAG)
ENDIF
8440 FORMAT ('IP=', I2, ', IZ=', I2, ', I=', I2, ', J=', I2, ', K=', I2,
', ' TM =', E11.4, ', HT =', E11.4, ', P =', E11.4,
', )
END IF
C
IF (MOD (IT0, ITPNT) .EQ. 0) THEN
PEXIT=0.
FACT=32.174*DNREF1*UREF1*XREF1*XREF1
IF (NPROC .LE. 1) THEN
IZ = IZON
L  = IZS(IZ)
M  = JZS(IZ)
N  = KZS(IZ)
DO IPONT=1,3
IF (IPONT .EQ. 1) THEN
I=1
IP1=1
ELSE IF (IPONT .EQ. 2) THEN
I=17
IP1=1
ELSE
I=IZT(IZ)
IP1=0
ENDIF
END IF
FLOWT=0.
FLOWO=0.
FLOWF=0.
TMSUM=0.
DENSUM=0.
GAMSUM=0.
CPSUM=0.
XMSUM=0.
XPFLUX=0.
XMFLUX=0.
DO KK=1,NGAS
FMSUM(KK)=0.
ENDDO
DO 8430 J=2,JZT(IZ)-1
DO 8430 K=2,KZT(IZ)-1
IJK  = L+I+(J-1)*M+(K-1)*N
ij1 = ijk+IP1
DX2  = .5*(xc95(ij1+m)-xc95(ij1)+xc95(ij1+n+m)-xc95(ij1+n))
DY2  = .5*(yc95(ij1+m)-yc95(ij1)+yc95(ij1+n+m)-yc95(ij1+n))
DZ2  = .5*(zc95(ij1+m)-zc95(ij1)+zc95(ij1+n+m)-zc95(ij1+n))
DX3  = .5*(xc95(ij1+n)-xc95(ij1)+xc95(ij1+n+m)-xc95(ij1+m))
DY3  = .5*(yc95(ij1+n)-yc95(ij1)+yc95(ij1+n+m)-yc95(ij1+m))
DZ3  = .5*(zc95(ij1+n)-zc95(ij1)+zc95(ij1+n+m)-zc95(ij1+m))
CXQ  =          (DY2*DZ3-DY3*DZ2)
CYQ  =          -(DX2*DZ3-DX3*DZ2)
CZQ  =          (DX2*DY3-DX3*DY2)
P1  = (DEN(ijk))*(cxq*u(ijk)+cyq*v(ijk)+czq*w(ijk))
XAREA=SQRT (CXQ*CXQ+CYQ*CYQ+CZQ*CZQ)
XMFLUX=XMFLUX+P1*U(IJK)
XPFLUX=XPFLUX+(P(IJK)-PEXIT)*XAREA
ENDDO
DO 8430 J=2,JZT(IZ)-1
DO 8430 K=2,KZT(IZ)-1
IJK  = L+I+(J-1)*M+(K-1)*N

8440 FMTMN, 1, NSPM, ITIDS(1), MTAG
8440 FMTMX, 1, NSPM, ITIDS(1), MTAG
8440 FMFMX, 1, NSPM, ITIDS(1), MTAG
FLOWT = FLOWT + P1*FACT
FM01 = FM(IJK,1)*8./9. + FM(IJK,2) + FM(IJK,3)*4./7. + FM(IJK,4)*8./11. + FM(IJK,6) + FM(IJK,8)*16./17.

FLOWO = FLOWO + P1*FACT*FM01
FMH1 = 1. - FM01
FLOWF = FLOWF + P1*FACT*FMH1
TMSUM = TMSUM + TM(IJK)*P1
DENSUM = DENSUM + DEN(IJK)*P1
GAMSUM = GAMSUM + GAMB(IJK)*P1
CPSUM = CPSUM + CPB(IJK)*P1
XMSUM = XMSUM + P1
DO KK = 1, NGAS
   FMSUM(KK) = FMSUM(KK) + FM(IJK, KK)*P1
ENDDO

CONTINUE

TMAVE = TMSUM*TREF1/AMAX1(1.E-20, XMSUM)
DENAVE = DENSUM*DNREF1*32.174/AMAX1(1.E-20, XMSUM)
GAMAVE = GAMSUM/GAMBC/AMAX1(1.E-20, XMSUM)
CPAVE = CPSUM/CPBARC/AMAX1(1.E-20, XMSUM)
FMS = 0.
DO KK = 1, NGAS
   FMSUM(KK) = FMSUM(KK)/AMAX1(1.E-20, XMSUM)
   FMS = FMS + FMSUM(KK)
ENDDO

FMDUM = 0.
DO KK = 1, NGAS
   FMSUM(KK) = FMSUM(KK)/FMS
   FMDUM = FMDUM + FMSUM(KK)*WTMOLE(KK)
ENDDO

OFRATE = FLOWO/AMAX1(1.E-20, FLOWF)
OFRATE = AMAX1(1.E-20, OFRATE)
WTAVE = 1./AMAX1(1.E-20, FMDUM)
XISPP = UREF1*XPFLUX/XMSUM/32.174
XISPM = UREF1*XMFFLUX/XMSUM/32.174
XISP = XISPP + XISPM
IF (IPONT .EQ. 1) THEN
   WRITE (6, 8432) FLOWT, FLOWO, FLOWF
   WRITE (46, 525) DENAVE, TMAVE, GAMAVE, CPAVE, WTAVE, OFRATE
   WRITE (46, 526) (FMSUM(KK), KK = 1, NGAS)
ELSE IF (IPONT .EQ. 2) THEN
   WRITE (6, 8433) FLOWT, FLOWO, FLOWF
   WRITE (46, 527) DENAVE, TMAVE, GAMAVE, CPAVE, WTAVE, OFRATE
   WRITE (46, 526) (FMSUM(KK), KK = 1, NGAS)
   WRITE (46, 8953) XISPP, XISPM, XISP
ELSE
   WRITE (6, 8434) FLOWT, FLOWO, FLOWF
   WRITE (46, 528) DENAVE, TMAVE, GAMAVE, CPAVE, WTAVE, OFRATE
   WRITE (46, 526) (FMSUM(KK), KK = 1, NGAS)
   WRITE (46, 8954) XISPP, XISPM, XISP
ENDIF
END DO
ELSE
MTAG = 89000
IF (IPROC .LE. 1) THEN
   DO IP = 2, NPROC
      DO IPONT = 1, 3
         CALL RECVNB3 (FLOWT, FLOWO, FLOWF, ITIDS(IP), MTAG)
         CALL RECVNB3 (DENAVE, TMAVE, GAMAVE, ITIDS(IP), MTAG)
      END DO
   END DO
ELSE
   IF (IPONT .EQ. 1) THEN
      CALL RECVNB3 (FLOWT, FLOWO, FLOWF, ITIDS(IP), MTAG)
      CALL RECVNB3 (DENAVE, TMAVE, GAMAVE, ITIDS(IP), MTAG)
   ELSE
      WRITE (6, 8434) FLOWT, FLOWO, FLOWF
      WRITE (46, 528) DENAVE, TMAVE, GAMAVE, CPAVE, WTAVE, OFRATE
      WRITE (46, 526) (FMSUM(KK), KK = 1, NGAS)
      WRITE (46, 8954) XISPP, XISPM, XISP
   ENDIF
ENDIF
END
CALL RECVNB3(CPAVE,WTAVE,OFRATE,ITIDS(IP),MTAG)
CALL RECVNB3(XISPP,XISPM,XISP,ITIDS(IP),MTAG)
CALL RECVNB(FMSUM,1,NSPM,ITIDS(IP),MTAG)
IF(IPONT .EQ. 1) THEN
  WRITE(6,8432) FLOWT,FLOWO,FLOWF
  WRITE(46,525)DENAVE,TMAVE,GAMAVE,CPAVE,WTAVE,OFRATE
  WRITE(46,526) (FMSUM(KK),KK=1,NGAS)
ELSE IF(IPONT .EQ. 2) THEN
  WRITE(6,8433) FLOWT,FLOWO,FLOWF
  WRITE(46,527)DENAVE,TMAVE,GAMAVE,CPAVE,WTAVE,OFRATE
  WRITE(46,526) (FMSUM(KK),KK=1,NGAS)
  WRITE(46,8953)XISPP,XISPM,XISP
ELSE
  WRITE(6,8434) FLOWT,FLOWO,FLOWF
  WRITE(46,528)DENAVE,TMAVE,GAMAVE,CPAVE,WTAVE,OFRATE
  WRITE(46,526) (FMSUM(KK),KK=1,NGAS)
  WRITE(46,8954)XISPP,XISPM,XISP
ENDIF
ENDDO
ENDDO
ELSE
  IZ = IZON
  L = IZS(IZ)
  M = JZS(IZ)
  N = KZS(IZ)
DO IPONT=1,3
  IF(IPONT .EQ. 1) THEN
    I=1
    IP1=1
  ELSE IF(IPONT .EQ. 2) THEN
    I=17
    IP1=1
  ELSE
    I=IZT(IZ)
    IP1=0
  ENDIF
  FLOWT=0.
  FLOWO=0.
  FLOWF=0.
  TMSUM=0.
  DENSUM=0.
  GAMSUM=0.
  CPSUM=0.
  XMSUM=0.
  XPFLUX=0.
  XMFLUX=0.
  XISPP=0.
  XISPM=0.
  XISP=0.
  DO KK=1,NGAS
    FMSUM(KK)=0.
  ENDDO
  DO 8431 J=2,JZT(IZ)-1
    DO 8431 K=2,KZT(IZ)-1
      IJK=L+I+(J-1)*M+(K-1)*N
      ij1=ijk+IP1
      DX2=.5*(xc95(ij1+m)-xc95(ij1)+xc95(ij1+n+m)-xc95(ij1+n))
      DY2=.5*(yc95(ij1+m)-yc95(ij1)+yc95(ij1+n+m)-yc95(ij1+n))
DZ2 = 0.5*(zc95(ij1+n)-zc95(ij1)+zc95(ij1+n+m)-zc95(ij1+m))
DX3 = 0.5*(xc95(ij1+n)-xc95(ij1)+xc95(ij1+n+m)-xc95(ij1+m))
DY3 = 0.5*(yc95(ij1+n)-yc95(ij1)+yc95(ij1+n+m)-yc95(ij1+m))
DZ3 = 0.5*(zc95(ij1+n)-zc95(ij1)+zc95(ij1+n+m)-zc95(ij1+m))

CXQ = (DY2*DZ3-DY3*DZ2)
CYQ = -(DX2*DZ3-DX3*DZ2)
CZQ = (DX2*DY3-DX3*DY2)
P1 = (DEN(ijk))*(cxq*u(ijk)+cyq*v(ijk)+czq*w(ijk))

XAREA = SQRT(CXQ*CXQ+CYQ*CYQ+CZQ*CZQ)
XMFLUX = XMFLUX+P1*U(IJK)
XPFLUX = XPFLUX+(P(IJK)-PEXIT)*XAREA
FLOWT = FLOWT+P1*FACT
FM01 = FM(IJK,1)*8./9.+FM(IJK,2)+FM(IJK,3)*4./7.+FM(IJK,4)*8./11.+FM(IJK,6)+FM(IJK,8)*16./17.
FLOWO = FLOWO+P1*FACT*FM01

FMSUM = FMSUM+FM(IJK,1) + FM(IJK,2) + FM(IJK,3) + FM(IJK,4) + FM(IJK,6) + FM(IJK,8)

FMH1 = 1. - FM01
FLOWF = FLOWF+P1*FACT*FMH1
TMSUM = TMSUM+TM(IJK)*P1
DENSUM = DENSUM+DEN(IJK)*P1
GAMSUM = GAMSUM+GAMB(IJK)*P1
CPSUM = CPSUM+CPBC(IJK)*P1
XMSUM = XMSUM+P1

DO KK=1,NGAS
    FMSUM(KK) = FMSUM(KK) + FM(IJK,KK) * P1
ENDDO

8431 CONTINUE
TMAVE = TMSUM*TREF1/AMAX1(1.E-20,XMSUM)
DENAVE = DENSUM*DNREF1*32.174/AMAX1(1.E-20,XMSUM)
GAMAVE = GAMSUM/GAMBC/AMAX1(1.E-20,XMSUM)
CPAVE = CPSUM/CPBARC/AMAX1(1.E-20,XMSUM)
FMS = 0.

DO KK=1,NGAS
    FMSUM(KK) = FMSUM(KK) / AMAX1(1.E-20,XMSUM)
FMS = FMS+FMSUM(KK)
ENDDO
FMDUM = 0.

DO KK=1,NGAS
    FMSUM(KK) = FMSUM(KK) / FMS
    FMDUM = FMDUM+FMSUM(KK)*WTMOLE(KK)
ENDDO

OFRATE = FLOWO/AMAX1(1.E-20,FLOWF)
OFRATE = AMAX1(1.E-20,OF RATE)
WTAVE = 1./AMAX1(1.E-20,FMDUM)
XISPP = UREF1*XPFLUX/XMSUM/32.174
XISPM = UREF1*XMFLUX/XMSUM/32.174
XISP = XISPP+XISPM

CALL SENDNB3(FLOWT,FLOWO,FLOWF,ITIDS(1),MTAG)
CALL SENDNB3(DENAVE,TMAVE,GAMAVE,ITIDS(1),MTAG)
CALL SENDNB3(CPAVE,WTAVE,OF RATE,ITIDS(1),MTAG)
CALL SENDNB3(XISPP,XISPM,XISP,ITIDS(1),MTAG)
CALL SENDNB(FMSUM,1,NSPM,ITIDS(1),MTAG)
END DO

ENDIF

8432 FORMAT(1X,'$ AT THE NOZZLE INLET, TOTAL ', & 'MASS FLOW RATE = ',E14.8,'/2X,'OXYDIZER MASS FLOW RATE =', & E14.8,'/2X,FUEL MASS FLOW RATE =',E14.8)
AT THE NOZZLE THROAT, TOTAL

MASS FLOW RATE = ',E14.8,'OXYDIZER MASS FLOW RATE =',
& E14.8,', FUEL MASS FLOW RATE =',E14.8)

AT THE NOZZLE EXIT, TOTAL

MASS FLOW RATE = ',E14.8,'OXYDIZER MASS FLOW RATE =',
& E14.8,', FUEL MASS FLOW RATE =',E14.8)

@ AT I = ',I3,', J = ',I3,', MACH NO. = ',F10.6)

MASS AVERAGE ',33(1H=),
& /1X,'AVERAGING FLOW PROPERTIES AT THE NOZZLE INLET',
& /1X,'DENSITY(lbm/ft^3) =',F12.6,'TEMPERATURE(Deg R) =',
& F12.6,'GAMMA =',F12.6,'CP =',F12.6,'MOLECULAR WEIGHT =',F12.6,'O/F RATIO =',F12.6)

MASS AVERAGE ',33(1H=),
& /1X,'AVERAGING FLOW PROPERTIES AT THE NOZZLE THROAT',
& /1X,'DENSITY(lbm/ft^3) =',F12.6,'TEMPERATURE(Deg R) =',
& F12.6,'GAMMA =',F12.6,'CP =',F12.6,'MOLECULAR WEIGHT =',F12.6,'O/F RATIO =',F12.6)

MASS AVERAGE ',33(1H=),
& /1X,'AVERAGING FLOW PROPERTIES AT THE NOZZLE EXIT',
& /1X,'DENSITY(lbm/ft^3) =',F12.6,'TEMPERATURE(Deg R) =',
& F12.6,'GAMMA =',F12.6,'CP =',F12.6,'MOLECULAR WEIGHT =',F12.6,'O/F RATIO =',F12.6)

Vacuum Isp@THROAT, (PRESSURE, MOMENTUM, TOTAL) =',2(1X,F6.2,1H,),1X,F6.2)

Vacuum Isp@EXIT, (PRESSURE, MOMENTUM, ',
& TOTAL) =',2(1X,F6.2,1H,),1X,F6.2)

MTAG = 89900
IF(IPROC .LE. 1) THEN
WRITE(59,8937)
IZ = 1
L = IZS(IZ)
M = JZS(IZ)
N = KZS(IZ)
DO J=JZT(IZ)-1,26,-1
FLCHCK = 0.
FLOWO = 0.
FLOWF = 0.
DO I=2,IZT(IZ)-1
DO K=2,KZT(IZ)-1
IJK = L+I+(J-1)*M+(K-1)*N
ij1 = ijk+M
DX1 = .5*(xc95(ij1+1)-xc95(ij1)+xc95(ij1+n+1)-xc95(ij1+n))
DY1 = .5*(yc95(ij1+1)-yc95(ij1)+yc95(ij1+n+1)-yc95(ij1+n))
DZ1 = .5*(zc95(ij1+1)-zc95(ij1)+zc95(ij1+n+1)-zc95(ij1+n))
DX3 = .5*(xc95(ij1+n)-xc95(ij1)+xc95(ij1+n+1)-xc95(ij1+1))
DY3 = .5*(yc95(ij1+n)-yc95(ij1)+yc95(ij1+n+1)-yc95(ij1+1))
DZ3 = .5*(zc95(ij1+n)-zc95(ij1)+zc95(ij1+n+1)-zc95(ij1+1))
EXQ = -(DY1*DX3-DX1*DZ3)
EYQ = (DX1*DX3-DY1*DZ1)
EZQ = -(DX1*DY3-DX3*DY1)
IF(J .EQ. JZT(IZ)-1) THEN
P1 = -DEN(IJ1)*EXQ*U(IJ1)+EYQ*V(IJ1)+EZQ*W(IJ1)
FMO= FM(IJ1,1)*8./9.+FM(IJ1,2)+FM(IJ1,3)*4./7.
& +FM(IJ1,4)*8./11.+FM(IJ1,6)+FM(IJ1,8)*16./17.
ELSE

END
DENC = 0.5*(DEN(IJK)+DEN(IJ1))
P1 = -0.5*DENC*(EXQ*(U(IJK)+U(IJ1))+EYQ*(V(IJK)+V(IJ1))
& +EZQ*(W(IJK)+W(IJ1)))
& FMO1 = FM(IJK,1)*8./9.+FM(IJK,2)+FM(IJK,3)*4./7.
& +FM(IJK,4)*8./11.+FM(IJK,6)+FM(IJK,8)*16./17.
& FMO2 = FM(IJ1,1)*8./9.+FM(IJ1,2)+FM(IJ1,3)*4./7.
& +FM(IJ1,4)*8./11.+FM(IJ1,6)+FM(IJ1,8)*16./17.
FMO = 0.5*(FMO1+FMO2)
ENDIF
FMH=1.-FMO
FLCHCK=FLCHCK+P1*FACT
FLOWO=FLOWO+P1*FACT*FMO
FLOWF=FLOWF+P1*FACT*FMH
ENDDO
ENDDO
OFRATE=FLOWO/AMAX1(1.E-20,FLOWF)
WRITE(59,8936)IPROC,I,J,FLCHCK,FLOWO,FLOWF,OFRATE
ENDDO
DO IP=2,NPROC
CALL RECVIT1(IZ2,ITIDS(IP),MTAG)
DO IZ=1,IZ2
CALL RECVIT2(II1,II2,ITIDS(IP),MTAG)
DO IZ=II1,II2
CALL RECVNB4(FLCHCK,FLOWO,FLOWF,OFRATE,ITIDS(IP),MTAG)
WRITE(59,8936)IP,IZ,I2,FLCHCK,FLOWO,FLOWF,OFRATE
ENDDO
ENDDO
ENDDO
C
ELSE
CALL SENDIT1(IZON,ITIDS(1),MTAG)
C-GC
IZ = 2
IZ = 1
L  = IZS(IZ)
M  = JZS(IZ)
N  = KZS(IZ)
CALL SENDIT2(1,IZT(IZ)-1,ITIDS(1),MTAG)
DO 8931 I=1,IZT(IZ)-1
FLCHCK = 0.
FLOWO  = 0.
FLOWF  = 0.
DO 8935 J=2,JZT(IZ)-1
DO 8935 K=2,KZT(IZ)-1
IJK= L+I+(J-1)*M+(K-1)*N
ij1= ijk+1
DX2= .5*(xc95(ij1+m)-xc95(ij1)+xc95(ij1+n+m)-xc95(ij1+n))
DY2= .5*(yc95(ij1+m)-yc95(ij1)+yc95(ij1+n+m)-yc95(ij1+n))
DZ2= .5*(zc95(ij1+m)-zc95(ij1)+zc95(ij1+n+m)-zc95(ij1+n))
DX3= .5*(xc95(ij1+n)-xc95(ij1)+xc95(ij1+n+m)-xc95(ij1+m))
DY3= .5*(yc95(ij1+n)-yc95(ij1)+yc95(ij1+n+m)-yc95(ij1+m))
DZ3= .5*(zc95(ij1+n)-zc95(ij1)+zc95(ij1+n+m)-zc95(ij1+m))
CXQ= (DY2*DZ3-DY3*DZ2)
CYQ= -(DX2*DZ3-DX3*DZ2)
CZQ= (DX2*DY3-DX3*DY2)
IF(I .EQ. 1) THEN
P1  = DEN(IJK)*(cxq*u(ijk)+cyq*v(ijk)+czq*w(ijk))
FMO = FM(IJK,1)*8./9.+FM(IJK,2)+FM(IJK,3)*4./7.
& +FM(IJK,4)*8./11.+FM(IJK,6)+FM(IJK,8)*16./17.
ENDIF

FMN2 = 0.
ELSE IF (I .EQ. IZT(IZ)-1) THEN
  P1  =  DEN(IJ1)*(cxq*u(ij1)+cyq*v(ij1)+czq*w(ij1))
  FMO =  FM(IJ1,1)*8./9.+FM(IJ1,2)+FM(IJ1,3)*4./7. 
       +FM(IJ1,4)*8./11.+FM(IJ1,6)+FM(IJ1,8)*16./17.
  FMN2 = 0.
ELSE
  DENC = 0.5*(DEN(IJK)+DEN(IJ1))
  P1  =  0.5*DENC*(cxq*(u(ijk)+u(ij1))+cyq*(v(ijk)+v(ij1))
           +czq*(w(ijk)+w(ij1)))
  & FMO1 =  FM(IJK,1)*8./9.+FM(IJK,2)+FM(IJK,3)*4./7.
         +FM(IJK,4)*8./11.+FM(IJK,6)+FM(IJK,8)*16./17.
  & FMO2 =  FM(IJ1,1)*8./9.+FM(IJ1,2)+FM(IJ1,3)*4./7.
         +FM(IJ1,4)*8./11.+FM(IJ1,6)+FM(IJ1,8)*16./17.
  FMO =  0.5*(FMO1+FMO2)
  FMN2 = 0.
ENDIF
FMH=1.-FMO-FMN2
FLCHCK=FLCHCK+P1*FACT
FLOWO=FLOWO+P1*FACT*FMO
FLOWF=FLOWF+P1*FACT*FMH
8935 CONTINUE
OFRATE=FLOWO/AMAX1(1.E-20,FLOWF)
CALL SENDNB4(FLCHCK,FLOWO,FLOWF,OFRATE,ITIDS(1),MTAG)
8931 CONTINUE
END IF
END IF
REWIND(59)
REWIND(46)
REWIND(36)
8936 FORMAT(3(3X,I3),4(5X,F10.5))
8937 FORMAT(4X,'IP',2X,'IZON',4X,'I',6X,'TOTAL FLOW',5X,'OXYGEN FLO',
 & 5X,'FUEL FLOW',8X,'O/F',/3X,'==',2X,'====',3X,'===',
 & 4(5X,'=========='))
3. List of Sample Grid Generator

C------ Grid Generation for Vortex Engine test case #2
PARAMETER (NI = 51, NJ = 51, NK=61, NZ = 2, NSP = 11)
DIMENSION IZS(NZ),IZT(NZ),JZS(NZ),JZT(NZ),KZS(NZ),KZT(NZ)
DIMENSION X(NI,NJ,NK,NZ),Y(NI,NJ,NK,NZ),Z(NI,NJ,NK,NZ),
  U(NI,NJ,NK,NZ),V(NI,NJ,NK,NZ),W(NI,NJ,NK,NZ),
  P(NI,NJ,NK,NZ),DEN(NI,NJ,NK,NZ),TM(NI,NJ,NK,NK,NZ),
  DK(NI,NJ,NK,NZ),DE(NI,NJ,NK,NZ),FM(NI,NJ,NK,NSP,NZ),
  FMFG(NSP),FMOX(NSP),FMIN(NSP),WTM(NSP)
DIMENSION XX(100,100),RR(100,100),XZ(100,100),RZ(100,100)
DIMENSION X3D(NI,NJ,NK),Y3D(NI,NJ,NK),Z3D(NI,NJ,NK)

C
  IPLOT=0
  IZON=2
  NGAS=11
  PI=4.*ATAN(1.)
  IF (IZON .GT. NZ) THEN
    WRITE(6,901) IZON,NZ
    STOP
  ENDIF

  C------- Injector hole locations
  IH1=1
  IH2=4
  IH3=13
  IH4=16
  IH5=28
  IH6=31
  IH7=38
  IH8=41
  ID1=1
  ID2=4
  ID3=11
  ID4=14
  ID5=26
  ID6=29
  ID7=38
  ID8=41
  CALL IE4(IC1,IC2,IC3,IC4,IH1,IH2,IH3,IH4)
  CALL IE4(IC5,IC6,IC7,IC8,IH5,IH6,IH7,IH8)
  JC1=1
  JC2=26
  JC3=36
  JC4=41
  KC1=1
  KC2=7
  KC3=13
  KC4=31
  KC5=37
  KC6=43
  KC7=61

C
  IZS(1)=IC1
IZT(1)=IC8
JZS(1)=JC1
JZT(1)=JC4
KZS(1)=KC1
KZT(1)=KC7

C------- Nozzle grid location
IN1=1
IN2=17
IN3=20
IN4=47
JN1=1
JN2=JC2
KN1=1
KN2=KC4
KN3=KC7
IZS(2)=IN1
IZT(2)=IN4
JZS(2)=JN1
JZT(2)=JN2
KZS(2)=KN1
KZT(2)=KN3

C
C reference conditions
PCHAMB=1000.
AMC=0.2
UREF=228.323598
DNREF=0.0022708*32.714
TREF=540.
QREF=DNREF*UREF*UREF/32.174
GAMA=1.3985
PREF=PCHAMB*144./QREF
XREF=1.
C------ Injector and chamber geometry definition
RSINGU=1.E-6
RCHAMB=1.176/XREF
XCHAMB=0.96/XREF
TCHAMB=22.5*PI/180.
RCORN1=0.125/XREF
ANGCR1=45.*PI/180.
RCORN2=0.125/XREF
ANGCR2=45.*PI/180.
RCTHRT=0.25/XREF
ANGTH1=90.*PI/180.
ANGTH2=15.*PI/180.
RTHRT=0.48/XREF
XLTHRT=0.375/XREF ! distance between chamber end and the throat
XTHRT=XCHAMB+XLTHRT
RTHRTD=RTHRT+RCTHRT*(1.-COS(ANGTH2)) ! at the end of curved throat
XTHRTD=XTHRT+RCTHRT*SIN(ANGTH2) ! at the end of curved throat
XLNOZ=2./XREF ! distance between the throat and the nozzle end
XNOZ=XTHRT+XLNOZ
RNOZ=RTHRTD+(XLNOZ-XTHRTD+XTHRT)*TAN(ANGTH2)
RRPINJ=0.02/XREF
XRPINJ=0.43/XREF
ARPINJ=25.*PI/180.
TRPINJ=ACOS((RCHAMB-2.*RRPINJ)/RCHAMB)
ROXINJ=0.03/XREF
XOXINJ=0.65/XREF
AOXINJ=11.*PI/180.
TOXINJ=ACOS((RCHAMB-2.*ROXINJ)/RCHAMB)

C
X0=0.
Y0=0.
Z0=0.
X1=X0
Y1=Y0+RCHAMB-RCORN1
X2=X0+RCORN1
Y2=Y0+RCHAMB
X3=X0+XRPINJ-RRPINJ
Y3=Y0+RCHAMB
X4=X0+XRPINJ+RRPINJ
Y4=Y0+RCHAMB
X5=X0+XOXINJ-ROXINJ
Y5=Y0+RCHAMB
X6=X0+XOXINJ+ROXINJ
Y6=Y0+RCHAMB
X7=X0+XCHAMB
Y7=Y0+RCHAMB
X8=X0+XCHAMB+RCORN2*SIN(ANGCR2)
Y8=Y0+RCHAMB-RCORN2*(1.-COS(ANGCR2))
X9=X0+XCHAMB+RCORN2
Y9=Y0+RCHAMB-RCORN2
X10=X9
Y10=Y0+RTHRT+RCTHRT
ZA0=0.
ZA1=ZA0+TCHAMB-TOXINJ
ZA2=ZA0+TCHAMB-TRPINJ
ZA3=ZA0+TCHAMB
ZA4=ZA0+2.*TCHAMB-TOXINJ
ZA5=ZA0+2.*TCHAMB-TRPINJ
ZA6=ZA0+2.*TCHAMB
C------ Nozzle geometry definition
XCINTF=X0+2.0/XREF
YCINTF=Y0
YYI=Y10-YCINTF
XXI=XCINTF-X10
RINTF=SQRT(XXI*XXI+YYI*YYI)
AINTF=ATAN2(YYI,XXI)
X11=X0+XTHRT
Y11=Y0+RTHRT
X12=X0+XTHRTD
Y12=Y0+RTHRTD
X13=X0+XNOZ
Y13=Y0+RNOZ
X14=X0+XNOZ
Y14=Y0
X15=X0+XTHRT
Y15=Y0
X16=X0+XCINTF-RINTF
Y16=Y0
WRITE(9,*) X0,Y0
WRITE(9,*) X1,Y1
WRITE(9,*) X2,Y2
WRITE(9,*) X3,Y3
WRITE(9,*) X4,Y4
WRITE(9,*) X5,Y5
WRITE(9,*) X6,Y6
WRITE(9,*) X7,Y7
WRITE(9,*) X8,Y8
WRITE(9,*) X9,Y9
WRITE(9,*) X10,Y10
WRITE(9,*) X11,Y11
WRITE(9,*) X12,Y12
WRITE(9,*) X13,Y13
WRITE(9,*) X14,Y14
WRITE(9,*) X15,Y15
WRITE(9,*) X16,Y16

C        Fuel (RP-1) inlet conditions
C
AREAFG=PI*RRPINJ*RRPINJ/144.*1.2727
FLOFG=1.15/16.
DNFG=46.6874
UFG=FLOFG/DNFG/AREAFG/UREF
TFG=540./TREF
DKFG=0.015*UFG**2
DEFG=0.09*DKFG**1.5/(2.*RRPINJ)
DO KK=1,NGAS
   FMFG(KK)=0.
ENDDO
FMFG(10)=1.0

C        Oxidizer (LOX) inlet conditions
C
AREAOX=PI*ROXINJ*ROXINJ/144.*1.273
FLOOX=3./16.
DNOX=65.6421
UOX=FLOOX/DNOX/AREAOX/UREF
TOX=200./TREF
DKOX=0.015*UOX**2
DEOX=0.09*DKOX**1.5/(2.*ROXINJ)
DO KK=1,NGAS
   FMOX(KK)=0.
ENDDO
FMOX(2)=1.0
DNFG=DNFG/DNREF
DNOX=DNOX/DNREF

C        CHAMBER INITIAL CONDITIONS (combustion gas of OF ratio = 2.6)
C
DO KK=1,NGAS
   FMIN(KK)=0.
ENDDO
FMIN(1)=0.257
FMIN(2)=0.0243
FMIN(3)=0.3804
FMIN(4)=0.2768
FMIN(5)=0.0074
FMIN(6)=0.008
FMIN(7)=0.0012
FMIN(8)=0.0449
WTM(1)=18.01520
WTM(2)=31.99880
WTM(3)=28.01040
WTM(4)=44.00980  
WTM(5)=2.01580  
WTM(6)=15.99940  
WTM(7)=1.00790  
WTM(8)=17.00730  
WTM(9)=28.05360  
WTM(10)=173.93230  
WTM(11)=1177.24600  
WTDUM=0.  
DO KK=1,NGAS  
  WTDUM=WTDUM+FMIN(KK)/WTM(KK)  
ENDDO  
WTMOL=1./WTDUM  
TCH=3600.*1.8  
DNIN=PCHAMB*144.*WTMOL/1545./TCH  
FLOTOT=FLOOX+FLOFG  
AREACH=PI*RCHAMB*RCHAMB/16./144.  
ARETOT=PI*Y10*Y10/16./144.  
UCH=FLOTOT/DNIN/AREACH/UREF  
UIN=FLOTOT/DNIN/ARETOT/UREF  
TMIN=TCH/TREF  
DNIN=DNIN/DNREF  
DKIN=0.015*UIN**2  
DEIN=0.09*DKIN**1.5/(0.06*RTHRT)  
WRITE(6,*)'** CHAMBER INITIAL FLOW CONDITIONS:'  
WRITE(6,*)'Pressure (psi) =',PREF*QREF/144.  
WRITE(6,*)'Temperature (deg R) =',TMIN*TREF  
WRITE(6,*)'Density (lb/ft^3) =',DNIN*DNREF  
WRITE(6,*)'Mean Molecular Weight =',WTMOL  
WRITE(6,*)'Velocity (ft/s) =',UCH*UREF  
WRITE(6,*)'** OXIDIZER INLET CONDITIONS:'  
WRITE(6,*)'Temperature (deg R) =',TOX*TREF  
WRITE(6,*)'Density (lb/ft^3) =',DNOX*DNREF  
WRITE(6,*)'Velocity (ft/s) =',UOX*UREF  
WRITE(6,*)'** FUEL INLET CONDITIONS:'  
WRITE(6,*)'Temperature (deg R) =',TFG*TREF  
WRITE(6,*)'Density (lb/ft^3) =',DNFG*DNREF  
WRITE(6,*)'Velocity (ft/s) =',UFG*UREF  
C  
C------  CHAMBER SECTION  
Y0=RSINGU  
I=IC1  
DY=Y1-Y0  
DTY=1.2  
DO J=JC1,JC3  
  CALL SPCNTL(J,JC1,JC3,3,DTY,FACT)  
  XX(I,J)=X0  
  RR(I,J)=Y0+DY*FACT  
ENDDO  
DTY=1.01  
DO J=JC3+1,JC4  
  CALL SPCNTL(J,JC3,JC4,2,DTY,FACT)  
  ANG=ANGCR1*FACT  
  XX(I,J)=X0+RCORN1*(1.-COS(ANG))  
  RR(I,J)=RR(I,JC3)+RCORN1*SIN(ANG)  
ENDDO  
C  
79
C
I=IC8
DTY=1.1
DO J=JC1,JC2
   CALL SPCNTL(J,JC1,JC2,3,DTY,FACT)
   ANG=AINTF*FACT
   XX(I,J)=X16+RINTF*(1.-COS(ANG))
   RR(I,J)=Y0+RINTF*SIN(ANG)
ENDDO
XX(I,JC2)=X10
RR(I,JC2)=Y10
DTY=1.1
DXX=X9-X10
DYY=Y9-Y10
DO J=JC2+1,JC3
   CALL SPCNTL(J,JC2,JC3,3,DTY,FACT)
   XX(I,J)=X10+DXX*FACT
   RR(I,J)=Y10+DYY*FACT
ENDDO
DTY=1.01
DO J=JC3+1,JC4
   CALL SPCNTL(J,JC3,JC4,2,DTY,FACT)
   ANG=ANGCR2*FACT
   XX(I,J)=X9-RCORN2*(1.-COS(ANG))
   RR(I,J)=RR(I,JC3)+RCORN2*SIN(ANG)
ENDDO
C
J=JC1
DTX=1.0
DXX=XX(IC8,J)-XX(IC1,J)
DO I=IC1+1,IC8-1
   CALL SPCNTL(I,IC1,IC8,3,DTX,FACT)
   XX(I,J)=XX(IC1,J)+DXX*FACT
   RR(I,J)=Y0
ENDDO
C
J=JC4
DTX=1.1
DO I=IC1+1,IC2
   CALL SPCNTL(I,IC1,IC2,1,DTX,FACT)
   ANG=ANGCR1*(1.+FACT)
   XX(I,J)=X0+RCORN1*(1.-COS(ANG))
   RR(I,J)=Y1+RCORN1*SIN(ANG)
ENDDO
XX(IC2,J)=X2
RR(IC2,J)=Y2
C
DTX=1.02
DXX=X3-X2
DYY=Y3-Y2
DO I=IC2+1,IC3
   CALL SPCNTL(I,IC2,IC3,2,DTX,FACT)
   XX(I,J)=X2+DXX*FACT
   RR(I,J)=Y2+DYY*FACT
ENDDO
C
DTX=1.02
DXX=X4-X3
DYY=Y4-Y3
DO I=IC3+1,IC4
   CALL SPCNTL(I,IC3,IC4,3,DTX,FACT)
   XX(I,J)=X3+DXX*FACT
   RR(I,J)=Y3+DYY*FACT
ENDDO
C
DTX=1.0
DXX=X5-X4
DYY=Y5-Y4
DO I=IC4+1,IC5
   CALL SPCNTL(I,IC4,IC5,3,DTX,FACT)
   XX(I,J)=X4+DXX*FACT
   RR(I,J)=Y4+DYY*FACT
ENDDO
C
DTX=1.05
DXX=X6-X5
DYY=Y6-Y5
DO I=IC5+1,IC6
   CALL SPCNTL(I,IC5,IC6,3,DTX,FACT)
   XX(I,J)=X5+DXX*FACT
   RR(I,J)=Y5+DYY*FACT
ENDDO
C
DTX=1.1
DXX=X7-X6
DYY=Y7-Y6
DO I=IC6+1,IC7
   CALL SPCNTL(I,IC6,IC7,1,DTX,FACT)
   XX(I,J)=X6+DXX*FACT
   RR(I,J)=Y6+DYY*FACT
ENDDO
C
DTX=1.1
DO I=IC7+1,IC8-1
   CALL SPCNTL(I,IC7,IC8,2,DTX,FACT)
   ANG=ANGCR2*FACT
   XX(I,J)=X7+RCORN2*SIN(ANG)
   RR(I,J)=Y7-RCORN2*(1.-COS(ANG))
ENDDO
C
C------  Interpolate grid in the X-R plane
CALL TRANSF(IC1,IC8,JC1,JC4,XX,RR)
C------  relocate the corner points
DXX=XX(IC1+2,JC4-2)-XX(IC1,JC4)
DYY=RR(IC1+2,JC4-2)-RR(IC1,JC4)
XX(IC1+1,JC4-1)=XX(IC1,JC4)+0.4*DXX
RR(IC1+1,JC4-1)=RR(IC1,JC4)+0.4*DYY
DXX=XX(IC8-2,JC4-2)-XX(IC8,JC4)
DYY=RR(IC8-2,JC4-2)-RR(IC8,JC4)
XX(IC8-1,JC4-1)=XX(IC8,JC4)+0.4*DXX
RR(IC8-1,JC4-1)=RR(IC8,JC4)+0.4*DYY
C
CALL IE4(IC1,IC2,IC3,IC4,ID1,ID2,ID3,ID4)
CALL IE4(IC5,IC6,IC7,IC8,ID5,ID6,ID7,ID8)
X3=X9-XOXINJ-ROXINJ
Y3=Y0+RCHAMB
FDNS-RFV/PVM

X4 = X9 - XOXINJ + ROXINJ
Y4 = Y0 + RCHAMB
X5 = X9 - XRPINJ - RRPINJ
Y5 = Y0 + RCHAMB
X6 = X9 - XRPINJ + RRPINJ
Y6 = Y0 + RCHAMB

C
I = IC1
DY = Y1 - Y0
DTY = 1.2
DO J = JC1, JC3
  CALL SPCNTL(J, JC1, JC3, 3, DTY, FACT)
  XZ(I, J) = X0
  RZ(I, J) = Y0 + DY*FACT
ENDDO
DTY = 1.01
DO J = JC3 + 1, JC4
  CALL SPCNTL(J, JC3, JC4, 2, DTY, FACT)
  ANG = ANGCR1*FACT
  XZ(I, J) = X0 + RCORN1*(1. - COS(ANG))
  RZ(I, J) = RZ(I, JC3) + RCORN1*SIN(ANG)
ENDDO
C
I = IC8
DTY = 1.1
DO J = JC1, JC2
  CALL SPCNTL(J, JC1, JC2, 3, DTY, FACT)
  ANG = AINTF*FACT
  XZ(I, J) = X16 + RINTF*(1. - COS(ANG))
  RZ(I, J) = Y0 + RINTF*SIN(ANG)
ENDDO
XZ(I, JC2) = X10
RZ(I, JC2) = Y10
DTY = 1.1
DXX = X9 - X10
DYY = Y9 - Y10
DO J = JC2 + 1, JC3
  CALL SPCNTL(J, JC2, JC3, 3, DTY, FACT)
  XZ(I, J) = X10 + DXX*FACT
  RZ(I, J) = Y10 + DYY*FACT
ENDDO
DTY = 1.01
DO J = JC3 + 1, JC4
  CALL SPCNTL(J, JC3, JC4, 2, DTY, FACT)
  ANG = ANGCR2*FACT
  XZ(I, J) = X9 - RCORN2*(1. - COS(ANG))
  RZ(I, J) = RZ(I, JC3) + RCORN2*SIN(ANG)
ENDDO
C
J = JC1
DTX = 1.0
DXX = XZ(IC8, J) - XZ(IC1, J)
DO I = IC1 + 1, IC8 - 1
  CALL SPCNTL(I, IC1, IC8, 3, DTX, FACT)
  XZ(I, J) = XZ(IC1, J) + DXX*FACT
  RZ(I, J) = Y0
ENDDO
C
J=JC4
DTX=1.1
DO I=IC1+1,IC2
   CALL SPCNTL(I,IC1,IC2,1,DTX,FACT)
   ANG=ANGCR1*(1.+FACT)
   XZ(I,J)=X0+RCORN1*(1.-COS(ANG))
   RZ(I,J)=Y1+RCORN1*SIN(ANG)
ENDDO
XZ(IC2,J)=X2
RZ(IC2,J)=Y2

C
DTX=1.1
DXX=X3-X2
DYY=Y3-Y2
DO I=IC2+1,IC3
   CALL SPCNTL(I,IC2,IC3,2,DTX,FACT)
   XZ(I,J)=X2+DXX*FACT
   RZ(I,J)=Y2+DYY*FACT
ENDDO

C
DTX=1.05
DXX=X4-X3
DYY=Y4-Y3
DO I=IC3+1,IC4
   CALL SPCNTL(I,IC3,IC4,3,DTX,FACT)
   XZ(I,J)=X3+DXX*FACT
   RZ(I,J)=Y3+DYY*FACT
ENDDO

C
DTX=1.0
DXX=X5-X4
DYY=Y5-Y4
DO I=IC4+1,IC5
   CALL SPCNTL(I,IC4,IC5,3,DTX,FACT)
   XZ(I,J)=X4+DXX*FACT
   RZ(I,J)=Y4+DYY*FACT
ENDDO

C
DTX=1.02
DXX=X6-X5
DYY=Y6-Y5
DO I=IC5+1,IC6
   CALL SPCNTL(I,IC5,IC6,3,DTX,FACT)
   XZ(I,J)=X5+DXX*FACT
   RZ(I,J)=Y5+DYY*FACT
ENDDO

C
DTX=1.02
DXX=X7-X6
DYY=Y7-Y6
DO I=IC6+1,IC7
   CALL SPCNTL(I,IC6,IC7,1,DTX,FACT)
   XZ(I,J)=X6+DXX*FACT
   RZ(I,J)=Y6+DYY*FACT
ENDDO

C
DTX=1.1
DO I=IC7+1,IC8-1
CALL SPCNTL(I,IC7,IC8,2,DTX,FACT)  
ANG=ANGCR2*FACT  
XZ(I,J)=X7+RCORN2*SIN(ANG)  
RZ(I,J)=Y7-RCORN2*(1.-COS(ANG))  
ENDDO

C------ Interpolate grid in the X-R plane
CALL TRANSF(IC1,IC8,JC1,JC4,XZ,RZ)

C------ relocate the corner points
DXX=XZ(IC1+2,JC4-2)-XZ(IC1,JC4)  
DYY=RZ(IC1+2,JC4-2)-RZ(IC1,JC4)  
XZ(IC1+1,JC4-1)=XZ(IC1,JC4)+0.4*DXX  
RZ(IC1+1,JC4-1)=RZ(IC1,JC4)+0.4*DYY  
DXX=XZ(IC8-2,JC4-2)-XZ(IC8,JC4)  
DYY=RZ(IC8-2,JC4-2)-RZ(IC8,JC4)  
XZ(IC8-1,JC4-1)=XZ(IC8,JC4)+0.4*DXX  
RZ(IC8-1,JC4-1)=RZ(IC8,JC4)+0.4*DYY

C---- Generate 3-D grid for the chamber region
DTZ0=1.1  
DTZ1=1.1  
DTZ2=1.1  
DTZ3=1.1  
DTZ4=1.1

C
IZ=1  
K=KC1  
DO I=IC1,IC8  
DO J=JC1,JC4  
X(I,J,K,IZ)=XZ(I,J)  
Y(I,J,K,IZ)=RZ(I,J)  
Z(I,J,K,IZ)=0.  
ENDDO  
ENDDO  
K=KC4  
ANG=ZA3  
DO J=JC1,JC4  
DO I=IC1,IC8  
X(I,J,K,IZ)=XX(I,J)  
Y(I,J,K,IZ)=RR(I,J)*COS(ANG)  
Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)  
ENDDO  
ENDDO  
K=KC7  
ANG=ZA6  
DO J=JC1,JC4  
DO I=IC1,IC8  
X(I,J,K,IZ)=XZ(I,J)  
Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)  
Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)  
ENDDO  
ENDDO

C
DTZ=DTZ0  
DO K=KC1+1,KC4-1  
CALL SPCNTL(K,KC1,KC4,3,DTZ,FACTZ)  
ANG=ZA0+(ZA3-ZA0)*FACTZ  
DO J=JC1,JC2  
DO I=IC1,IC8  

84
X(I,J,K,IZ)=XZ(I,J)+(XX(I,J)-XZ(I,J))*FACTZ
RRR=RZ(I,J)+(RR(I,J)-RZ(I,J))*FACTZ
Y(I,J,K,IZ)=RRR*COS(ANG)
Z(I,J,K,IZ)=RRR*SIN(ANG)
ENDDO
ENDDO
ENDDO

DTZ=DTZ0
DO K=KC4+1,KC7-1
CALL SPCNTL(K,KC4,KC7,3,DTZ,FACTZ)
ANG=ZA3+(ZA6-ZA3)*FACTZ
DO J=JC1,JC2
DO I=IC1,IC8
X(I,J,K,IZ)=XX(I,J)+(XZ(I,J)-XX(I,J))*FACTZ
RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ
Y(I,J,K,IZ)=RRR*COS(ANG)
Z(I,J,K,IZ)=RRR*SIN(ANG)
ENDDO
ENDDO
ENDDO

C----- injector hole locations
J=JC4
DTZ=DTZ1
DO K=KC4+1,KC7-1
CALL SPCNTL(K,KC4,KC7,3,DTZ,FACTZ)
ANG=ZA3+(ZA6-ZA3)*FACTZ
DO J=JC4
DO I=IC1,IC8
X(I,J,K,IZ)=XX(I,J)+(XZ(I,J)-XX(I,J))*FACTZ
RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ
Y(I,J,K,IZ)=RRR*COS(ANG)
Z(I,J,K,IZ)=RRR*SIN(ANG)
ENDDO
ENDDO
ENDDO

DTZ=DTZ2
DO K=KC3+1,KC4-1
CALL SPCNTL(K,KC3,KC4,3,DTZ,FACTZ)
ANG=ZA2+(ZA3-ZA2)*FACTZ
DO I=IC1,IC2
X(I,J,K,IZ)=XX(I,J)
Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
ENDDO
ENDDO
ENDDO

DTZ=DTZ3
DO K=KC4+1,KC5
CALL SPCNTL(K,KC4,KC5,3,DTZ,FACTZ)
ANG=ZA3+(ZA4-ZA3)*FACTZ
DO I=IC1,IC2
X(I,J,K,IZ)=XX(I,J)+(XZ(I,J)-XX(I,J))*FACTZ
RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ
Y(I,J,K,IZ)=RRR*COS(ANG)
Z(I,J,K,IZ)=RRR*SIN(ANG)
ENDDO
ENDDO
ENDDO

DTZ=DTZ4
DO K=KC5+1,KC7-1
CALL SPCNTL(K,KC5,KC7,3,DTZ,FACTZ)
ANG=ZA4+(ZA6-ZA4)*FACTZ

85
DO I=IC1,IC2
X(I,J,K,IZ)=XZ(I,J)
Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)
Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)
ENDDO
ENDDO

CALL IE4(IC3,IC4,IC5,IC6,IC7,IC8,IC9,IC10,IC11,IC12)
DTZ=DTZ2
DO I=IC3,IC4
DO K=KC3,KC4-1
CALL SPCNTL(K,KC3,KC4,3,DTZ,FACTZ)
ANG=ZA2+(ZA3-ZA2)*FACTZ
Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
X(I,J,K,IZ)=X(I,J,KC4,IZ)
&               +(Z(I,J,K,IZ)-Z(I,J,KC4,IZ))*TAN(ARPINJ)
ENDDO
ENDDO

K=KC3
DTX=1.0
DXX=X(IC3,J,K,IZ)-X(IC2,J,K,IZ)
DO I=IC2+1,IC3-1
CALL SPCNTL(I,IC2,IC3,3,DTX,FACTX)
ANG=ZA2
Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
X(I,J,K,IZ)=X(IC2,J,K,IZ)+DXX*FACTX
ENDDO

DTZ=DTZ4
DO I=IC5,IC6
DO K=KC2,KC4-1
CALL SPCNTL(K,KC2,KC4,3,DTZ,FACTZ)
ANG=ZA2+(ZA3-ZA2)*FACTZ
Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
X(I,J,K,IZ)=X(I,J,KC2,IZ)+DZZ*(X(I,J,KC4,IZ)-X(I,J,KC2,IZ))
ENDDO
ENDDO

DTZ=DTZ2
DO I=IC2+1,IC3-1
CALL SPCNTL(I,IC2,IC3,3,DTZ,FACTX)
ANG=ZA2
Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
X(I,J,K,IZ)=X(I,J,KC2,IZ)+DZZ*(X(I,J,KC4,IZ)-X(I,J,KC2,IZ))
ENDDO
ENDDO

DTZ=DTZ4
DO I=IC5,IC6
DO K=KC2,KC4-1
CALL SPCNTL(K,KC2,KC4,3,DTZ,FACTZ)
ANG=ZA1+(ZA3-ZA1)*FACTZ
Y(I,J,K,I2)=RR(I,J)*COS(ANG)
Z(I,J,K,I2)=RR(I,J)*SIN(ANG)
X(I,J,K,I2)=X(I,J,KC4,I2)
&
   -(Z(I,J,K,I2)-Z(I,J,KC4,I2))*TAN(AOXINJ)
ENDDO

C

DTZ=DTZ3
DO K=KC1+1,KC2
   CALL SPCNTL(K,KC1,KC2,3,DTZ,FACTZ)
   ANG=ZA0+(ZA1-ZA0)*FACTZ
   DO I=IC7,IC8
      X(I,J,K,I2)=XZ(I,J)+(XX(I,J)-XZ(I,J))*FACTZ
      RRR=RZ(I,J)+(RR(I,J)-RZ(I,J))*FACTZ
      Y(I,J,K,I2)=RRR*COS(ANG)
      Z(I,J,K,I2)=RRR*SIN(ANG)
   ENDDO
ENDDO

DTZ=DTZ4
DO K=KC2+1,KC4-1
   CALL SPCNTL(K,KC2,KC4,3,DTZ,FACTZ)
   ANG=ZA1+(ZA3-ZA1)*FACTZ
   DO I=IC7,IC8
      X(I,J,K,I2)=XX(I,J)
      Y(I,J,K,I2)=RR(I,J)*COS(ANG)
      Z(I,J,K,I2)=RR(I,J)*SIN(ANG)
   ENDDO
ENDDO

DTZ=DTZ1
DO K=KC4+1,KC6
   CALL SPCNTL(K,KC4,KC6,3,DTZ,FACTZ)
   ANG=ZA3+(ZA5-ZA3)*FACTZ
   DO I=IC7,IC8
      X(I,J,K,I2)=XZ(I,J)+(XZ(I,J)-XX(I,J))*FACTZ
      RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ
      Y(I,J,K,I2)=RRR*COS(ANG)
      Z(I,J,K,I2)=RRR*SIN(ANG)
   ENDDO
ENDDO

DTZ=DTZ2
DO K=KC6+1,KC7-1
   CALL SPCNTL(K,KC6,KC7,3,DTZ,FACTZ)
   ANG=ZA5+(ZA6-ZA5)*FACTZ
   DO I=IC7,IC8
      X(I,J,K,I2)=XZ(I,J)
      Y(I,J,K,I2)=RZ(I,J)*COS(ANG)
      Z(I,J,K,I2)=RZ(I,J)*SIN(ANG)
   ENDDO
ENDDO

K=KC2
DXX=X(IC7,J,K,I2)-X(IC6,J,K,I2)
DO I=IC6+1,IC7-1
   CALL SPCNTL(I,IC6,IC7,3,DTX,FACTX)
   ANG=ZA1
Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
X(I,J,K,IZ)=X(IC6,J,K,IZ)+DXX*FACTX
ENDDO
C

DTZ=DTZ3
DO I=IC5,IC7-1
  DO K=KC1+1,KC2-1
    CALL SPCNTL(K,KC1,KC2,3,DTZ,FACTZ)
    ANG=ZA0+(ZA1-ZA0)*FACTZ
    RRR=RZ(I,J)+(RR(I,J)-RZ(I,J))*FACTZ
    Y(I,J,K,IZ)=RRR*COS(ANG)
    Z(I,J,K,IZ)=RRR*SIN(ANG)
    D2Z=(Z(I,J,K,IZ)-Z(I,J,KC1,IZ))/(Z(I,J,KC2,IZ)-Z(I,J,KC1,IZ))
    X(I,J,K,IZ)=X(I,J,KC1,IZ)+D2Z*(X(I,J,KC2,IZ)-X(I,J,KC1,IZ))
  ENDDO
ENDDO
DTZ=DTZ4
DO I=IC6+1,IC7-1
  DO K=KC2+1,KC4-1
    CALL SPCNTL(K,KC2,KC4,3,DTZ,FACTZ)
    ANG=ZA1+(ZA3-ZA1)*FACTZ
    Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
    Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
    X(I,J,K,IZ)=X(I,J,KC4,IZ)+D2Z*(X(I,J,KC2,IZ)-X(I,J,KC4,IZ))
  ENDDO
ENDDO
C---- Gap between injector holes
DTX=1.0
DO K=KC1+1,KC4-1
  ANG1=ATAN2(Z(IC4,J,K,IZ),Y(IC4,J,K,IZ))
  ANG2=ATAN2(Z(IC5,J,K,IZ),Y(IC5,J,K,IZ))
  DANG=ANG2-ANG1
  DXX=X(IC5,J,K,IZ)-X(IC4,J,K,IZ)
  DO I=IC4+1,IC5-1
    CALL SPCNTL(I,IC4,IC5,3,DTX,FACT)
    X(I,J,K,IZ)=X(IC4,J,K,IZ)+DXX*FACT
    ANG=ANG1+DANG*FACT
    Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
    Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
  ENDDO
ENDDO
C---- 2nd pair of unlike impinger
CALL IE4(IC3,IC4,IC5,IC6,ID3,ID4,ID5,ID6)
DTZ=DTZ2
DO I=IC3,IC4
  DO K=KC5,KC7-1
    CALL SPCNTL(K,KC5,KC7,3,DTZ,FACTZ)
    ANG=ZA4+(ZA6-ZA4)*FACTZ
    Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)
    Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)
    X(I,J,K,IZ)=X(I,J,KC7,IZ)
    Z(I,J,K,IZ)+=(Z(I,J,K,IZ)-Z(I,J,KC7,IZ))*TAN(AOxinj)
  ENDDO
ENDDO
C
K=KC5
DTX=1.0  
DXX=X(IC3,J,K,I2)-X(IC2,J,K,I2)  
DO I=IC2+1,IC3-1  
   CALL SPCNTL(I,IC2,IC3,3,DTX,FACTX)  
   ANG=ZA4  
   Y(I,J,K,I2)=RZ(I,J)*COS(ANG)  
   Z(I,J,K,I2)=RZ(I,J)*SIN(ANG)  
   X(I,J,K,I2)=X(IC2,J,K,I2)+DXX*FACTX  
ENDDO

C

DTZ=DTZ3  
DO I=IC2+1,IC4  
   DO K=KC4+1,KC5-1  
      CALL SPCNTL(K,KC4,KC5,3,DTZ,FACTZ)  
      ANG=ZA3+(ZA4-ZA3)*FACTZ  
      RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ  
      Y(I,J,K,I2)=RRR*COS(ANG)  
      Z(I,J,K,I2)=RRR*SIN(ANG)  
      X(I,J,K,I2)=X(I,J,KC4,I2)+DZZ*(X(I,J,KC5,I2)-X(I,J,KC4,I2))  
   ENDDO  
ENDDO

C

DTZ=DTZ4  
DO I=IC2+1,IC3-1  
   DO K=KC5+1,KC7-1  
      CALL SPCNTL(K,KC5,KC7,3,DTZ,FACTZ)  
      ANG=ZA4+(ZA6-ZA4)*FACTZ  
      Y(I,J,K,I2)=RZ(I,J)*COS(ANG)  
      Z(I,J,K,I2)=RZ(I,J)*SIN(ANG)  
      X(I,J,K,I2)=X(I,J,KC7,I2)+DZZ*(X(I,J,KC5,I2)-X(I,J,KC7,I2))  
   ENDDO  
ENDDO

C

DTZ=DTZ2  
DO I=IC5,IC6  
   DO K=KC6,KC7-1  
      CALL SPCNTL(K,KC6,KC7,3,DTZ,FACTZ)  
      ANG=ZA5+(ZA6-ZA5)*FACTZ  
      Y(I,J,K,I2)=RZ(I,J)*COS(ANG)  
      Z(I,J,K,I2)=RZ(I,J)*SIN(ANG)  
      X(I,J,K,I2)=X(I,J,KC7,I2)  
&               -(Z(I,J,K,I2)-Z(I,J,KC7,I2))*TAN(ARPINJ)  
   ENDDO  
ENDDO

C

K=KC6  
DTX=1.0  
DXX=X(IC7,J,K,I2)-X(IC6,J,K,I2)  
DO I=IC6+1,IC7-1  
   CALL SPCNTL(I,IC6,IC7,3,DTX,FACTX)  
   ANG=ZA5  
   Y(I,J,K,I2)=RZ(I,J)*COS(ANG)  
   Z(I,J,K,I2)=RZ(I,J)*SIN(ANG)  
   X(I,J,K,I2)=X(IC6,J,K,I2)+DXX*FACTX  
ENDDO

C

DTZ=DTZ1
DO I=IC5,IC7-1
  DO K=KC4+1,KC6-1
    CALL SPCNTL(K,KC4,KC6,3,DTZ,FACTZ)
    ANG=ZA3+(ZA5-ZA3)*FACTZ
    RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ
    Y(I,J,K,IZ)=RRR*COS(ANG)
    Z(I,J,K,IZ)=RRR*SIN(ANG)
    DZZ=(Z(I,J,K,IZ)-Z(I,J,KC4,IZ))/(Z(I,J,KC6,IZ)-Z(I,J,KC4,IZ))
    X(I,J,K,IZ)=X(I,J,KC4,IZ)+DZZ*(X(I,J,KC6,IZ)-X(I,J,KC4,IZ))
  ENDDO
ENDDO
DTZ=DTZ2
DO I=IC6+1,IC7-1
  DO K=KC6+1,KC7-1
    CALL SPCNTL(K,KC6,KC7,3,DTZ,FACTZ)
    ANG=ZA5+(ZA6-ZA5)*FACTZ
    Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)
    Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)
    X(I,J,K,IZ)=X(I,J,KC7,IZ)+DZZ*(X(I,J,KC6,IZ)-X(I,J,KC7,IZ))
  ENDDO
ENDDO
C---- Gap between injector holes
DTX=1.0
DO K=KC4+1,KC7-1
  ANG1=ATAN2(Z(IC4,J,K,IZ),Y(IC4,J,K,IZ))
  ANG2=ATAN2(Z(IC5,J,K,IZ),Y(IC5,J,K,IZ))
  DANG=ANG2-ANG1
  DXX=X(IC5,J,K,IZ)-X(IC4,J,K,IZ)
  DO I=IC4+1,IC5-1
    CALL SPCNTL(I,IC4,IC5,3,DTX,FACT)
    X(I,J,K,IZ)=X(IC4,J,K,IZ)+DXX*FACT
    ANG=ANG1+DANG*FACT
    Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)
    Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)
  ENDDO
ENDDO
C
I=IC1
DTZ=DTZ1
DO K=KC1+1,KC3
  CALL SPCNTL(K,KC1,KC3,3,DTZ,FACTZ)
  ANG=ZA0+(ZA2-ZA0)*FACTZ
  DO J=JC2+1,JC4-1
    X(I,J,K,IZ)=XZ(I,J)+(XX(I,J)-XZ(I,J))*FACTZ
    RRR=RR(I,J)+(RR(I,J)-RR(I,J))*FACTZ
    Y(I,J,K,IZ)=RRR*COS(ANG)
    Z(I,J,K,IZ)=RRR*SIN(ANG)
  ENDDO
ENDDO
DTZ=DTZ2
DO K=KC3+1,KC4-1
  CALL SPCNTL(K,KC3,KC4,3,DTZ,FACTZ)
  ANG=ZA2+(ZA3-ZA2)*FACTZ
  DO J=JC2+1,JC4-1
    X(I,J,K,IZ)=XX(I,J)
    Y(I,J,K,IZ)=RR(I,J)*COS(ANG)
    Z(I,J,K,IZ)=RR(I,J)*SIN(ANG)
  ENDDO
ENDDO
90
FDNS-RFV/PVM

ENDDO
ENDDO

DTZ=DTZ2

DO  K=KC4+1,KC5
    CALL  SPCNTL(K,KC4,KC5,3,DTZ,FACTZ)
    ANG=ZA3+(ZA4-ZA3)*FACTZ
    DO  J=JC2+1,JC4-1
        X(I,J,K,IZ)=XX(I,J)+(XZ(I,J)-XX(I,J))*FACTZ
        RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ
        Y(I,J,K,IZ)=RRR*COS(ANG)
        Z(I,J,K,IZ)=RRR*SIN(ANG)
    ENDDO
ENDDO

ENDDO

DTZ=DTZ1

DO  K=KC5+1,KC7-1
    CALL  SPCNTL(K,KC5,KC7,3,DTZ,FACTZ)
    ANG=ZA4+(ZA6-ZA4)*FACTZ
    DO  J=JC2+1,JC4-1
        X(I,J,K,IZ)=XZ(I,J)
        Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)
        Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)
    ENDDO
ENDDO


C

I=IC8

DTZ=DTZ3

DO  K=KC1+1,KC2
    CALL  SPCNTL(K,KC1,KC2,3,DTZ,FACTZ)
    ANG=ZA0+(ZA1-ZA0)*FACTZ
    DO  J=JC2+1,JC4-1
        X(I,J,K,IZ)=XZ(I,J)+(XX(I,J)-XZ(I,J))*FACTZ
        RRR=RR(I,J)+(RR(I,J)-RZ(I,J))*FACTZ
        Y(I,J,K,IZ)=RRR*COS(ANG)
        Z(I,J,K,IZ)=RRR*SIN(ANG)
    ENDDO
ENDDO


DO K=KC6+1,KC7-1
  CALL SPCNTL(K,KC6,KC7,3,DTZ,FACTZ)
  ANG=ZA5+(ZA6-ZA5)*FACTZ
  DO J=JC2+1,JC4-1
    X(I,J,K,IZ)=XZ(I,J)
    Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)
    Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)
  ENDDO
ENDDO

C
DO I=IC1,IC8
  DO K=KC1,KC7
    DO J=JC2,JC4
      X3D(I,J,K)=X(I,J,K,IZ)
      Y3D(I,J,K)=Y(I,J,K,IZ)
      Z3D(I,J,K)=Z(I,J,K,IZ)
    ENDDO
  ENDDO
ENDDO
ENDDO
CALL TRAN3D(IC1,IC8,JC2,JC4,KC1,KC4,X3D,Y3D,Z3D)
CALL TRAN3D(IC1,IC8,JC2,JC4,KC4,KC7,X3D,Y3D,Z3D)
DO I=IC1,IC8
  DO K=KC1,KC7
    DO J=JC2,JC4
      X(I,J,K,IZ)=X3D(I,J,K)
      Y(I,J,K,IZ)=Y3D(I,J,K)
      Z(I,J,K,IZ)=Z3D(I,J,K)
    ENDDO
  ENDDO
ENDDO
ENDDO

C---- Nozzle grid in 2-D plane
100 CONTINUE
I=IN1
  DO J=JN1,JN2
    XX(I,J)=XX(IC8,J)
    RR(I,J)=RR(IC8,J)
    XZ(I,J)=XZ(IC8,J)
    RZ(I,J)=RZ(IC8,J)
  ENDDO
J=JN1
  DTX=1.05
  DXX=X15-XX(IN1,J)
  DXZ=X15-XZ(IN1,J)
  DO I=IN1+1,IN2
    CALL SPCNTL(I,IN1,IN2,3,DTX,FACT)
    XX(I,J)=XX(IN1,J)+DXX*FACT
    RR(I,J)=Y0
    XZ(I,J)=XZ(IN1,J)+DXZ*FACT
    RZ(I,J)=Y0
  ENDDO
  DTX=1.4
  DXX=0.9*(X14-XX(IN2,J))
  DXZ=0.9*(X14-XZ(IN2,J))
  INN=IN4-3
  DO I=IN2+1,INN
    CALL SPCNTL(I,IN2,INN,1,DTX,FACT)
    XX(I,J)=XX(IN2,J)+DXX*FACT
  ENDDO
RR(I,J)=Y0
XZ(I,J)=XZ(IN2,J)+DXZ*FACT
RZ(I,J)=Y0
ENDDO

DTX=1.2
DXX=X14-XX(INN,J)
DXZ=X14-XZ(INN,J)
DO I=INN+1,IN4
   CALL SPCNTL(I,INN,IN4,2,DTX,FACT)
   XX(I,J)=XX(INN,J)+DXX*FACT
   RR(I,J)=Y0
   XZ(I,J)=XZ(INN,J)+DXZ*FACT
   RZ(I,J)=Y0
ENDDO

J=JN2
DTX=1.00
DO I=IN1+1,IN2
   CALL SPCNTL(I,IN1,IN2,3,DTX,FACT)
   ANG=ANGTH1*FACT
   XX(I,J)=XX(IN1,J)+RCTHRT*(1.-COS(ANG))
   RR(I,J)=RR(IN1,J)-RCTHRT*SIN(ANG)
   XZ(I,J)=XZ(IN1,J)+RCTHRT*(1.-COS(ANG))
   RZ(I,J)=RZ(IN1,J)-RCTHRT*SIN(ANG)
ENDDO

DTX=1.01
DO I=IN2+1,IN3
   CALL SPCNTL(I,IN2,IN3,1,DTX,FACT)
   ANG=ANGTH2*FACT
   XX(I,J)=XX(IN2,J)+RCTHRT*SIN(ANG)
   RR(I,J)=RR(IN2,J)+RCTHRT*(1.-COS(ANG))
   XZ(I,J)=XZ(IN2,J)+RCTHRT*SIN(ANG)
   RZ(I,J)=RZ(IN2,J)+RCTHRT*(1.-COS(ANG))
ENDDO

DTX=1.2
DXX=0.9*(X13-XX(IN3,J))
DYY=0.9*(Y13-RR(IN3,J))
DXZ=0.9*(X13-XZ(IN3,J))
DYZ=0.9*(Y13-RZ(IN3,J))
INN=IN4-3
DO I=IN3+1,INN
   CALL SPCNTL(I,IN3,INN,1,DTX,FACT)
   XX(I,J)=XX(IN3,J)+DXX*FACT
   RR(I,J)=RR(IN3,J)+DYY*FACT
   XZ(I,J)=XZ(IN3,J)+DXZ*FACT
   RZ(I,J)=RZ(IN3,J)+DYZ*FACT
ENDDO

DYY=Y13-RR(INN,J)
DYZ=Y13-RZ(INN,J)
DO I=INN+1,IN4
   XX(I,J)=XX(I,JN1)
   FACT=(XX(I,J)-XX(INN,JN1))/(XX(IN4,JN1)-XX(INN,JN1))
   RR(I,J)=RR(INN,J)+DYY*FACT
   XZ(I,J)=XZ(INN,JN1)
   FACT=(XZ(I,J)-XZ(INN,JN1))/(XZ(IN4,JN1)-XZ(INN,JN1))
   RZ(I,J)=RZ(INN,J)+DYZ*FACT
ENDDO
I=IN2
DTY=1.05
DXX = XX(I, JN2) - XX(I, JN1)
DYY = RR(I, JN2) - RR(I, JN1)
DXZ = XZ(I, JN2) - XZ(I, JN1)
DYZ = RZ(I, JN2) - RZ(I, JN1)
DO J = JN1 + 1, JN2 - 1
    CALL SPCNTL(J, JN1, JN2, 3, DTY, FACT)
    XX(I, J) = XX(I, JN1) + DXX * FACT
    RR(I, J) = RR(I, JN1) + DYY * FACT
    XZ(I, J) = XZ(I, JN1) + DXZ * FACT
    RZ(I, J) = RZ(I, JN1) + DYZ * FACT
ENDDO
I = IN4
DTY = 1.2
DXX = XX(I, JN2) - XX(I, JN1)
DYY = RR(I, JN2) - RR(I, JN1)
DXZ = XZ(I, JN2) - XZ(I, JN1)
DYZ = RZ(I, JN2) - RZ(I, JN1)
DO J = JN1 + 1, JN2 - 1
    CALL SPCNTL(J, JN1, JN2, 3, DTX, FACT)
    XX(I, J) = XX(I, JN1) + DXX * FACT
    RR(I, J) = RR(I, JN1) + DYY * FACT
    XZ(I, J) = XZ(I, JN1) + DXZ * FACT
    RZ(I, J) = RZ(I, JN1) + DYZ * FACT
ENDDO
CALL TRANSF(IN1, IN2, JN1, JN2, XX, RR)
CALL TRANSF(IN2, IN4, JN1, JN2, XX, RR)
CALL TRANSF(IN1, IN2, JN1, JN2, XZ, RZ)
CALL TRANSF(IN2, IN4, JN1, JN2, XZ, RZ)
C------- Generate 3-D grids for nozzle region
IZ = 2
I = IN1
DO J = JN1, JN2
    DO K = KN1, KN3
        X(I, J, K, IZ) = X(IC8, J, K, 1)
        Y(I, J, K, IZ) = Y(IC8, J, K, 1)
        Z(I, J, K, IZ) = Z(IC8, J, K, 1)
    ENDDO
ENDDO
ENDDO
C
K = KN1
DO I = IN1 + 1, IN4
    DO J = JN1, JN2
        X(I, J, K, IZ) = XZ(I, J)
        Y(I, J, K, IZ) = RZ(I, J)
        Z(I, J, K, IZ) = 0.
    ENDDO
ENDDO
ENDDO
K = KN2
ANG = ZA3
DO I = IN1 + 1, IN4
    DO J = JN1, JN2
        X(I, J, K, IZ) = XX(I, J)
        Y(I, J, K, IZ) = RR(I, J) * COS(ANG)
        Z(I, J, K, IZ) = RR(I, J) * SIN(ANG)
    ENDDO
ENDDO
ENDDO
K = KN3
ANG = ZA6
DO I=IN1+1,IN4
DO J=JN1,JN2
   X(I,J,K,IZ)=XZ(I,J)
   Y(I,J,K,IZ)=RZ(I,J)*COS(ANG)
   Z(I,J,K,IZ)=RZ(I,J)*SIN(ANG)
ENDDO
ENDDO

C

DTZ=DTZ0
DO K=KN1+1,KN2-1
   CALL SPCNTL(K,KN1,KN2,3,DTZ,FACTZ)
   ANG=ZA0+(ZA3-ZA0)*FACTZ
   DO I=IN1+1,IN4
      DO J=JN1,JN2
         X(I,J,K,IZ)=XZ(I,J)+(XX(I,J)-XZ(I,J))*FACTZ
         RRR=RZ(I,J)+(RR(I,J)-RZ(I,J))*FACTZ
         Y(I,J,K,IZ)=RRR*COS(ANG)
         Z(I,J,K,IZ)=RRR*SIN(ANG)
      ENDDO
   ENDDO
ENDDO

DO K=KN2+1,KN3-1
   CALL SPCNTL(K,KN2,KN3,3,DTZ,FACTZ)
   ANG=ZA3+(ZA6-ZA3)*FACTZ
   DO I=IN1+1,IN4
      DO J=JN1,JN2
         X(I,J,K,IZ)=XX(I,J)+(XZ(I,J)-XX(I,J))*FACTZ
         RRR=RR(I,J)+(RZ(I,J)-RR(I,J))*FACTZ
         Y(I,J,K,IZ)=RRR*COS(ANG)
         Z(I,J,K,IZ)=RRR*SIN(ANG)
      ENDDO
   ENDDO
ENDDO

C

C-------  SPECIFY INLET & INITIAL FLOW FIELD
IZ=1
DO 6000 I=1,IZT(IZ)
   DO 6000 K=1,KZT(IZ)
      DO 6100 J=1,JZT(IZ)
         U(I,J,K,IZ)=UCH
         IF(I .EQ. IZT(IZ)) U(I,J,K,IZ)=UIN
         V(I,J,K,IZ)=0.
         W(I,J,K,IZ)=0.
         P(I,J,K,IZ)=PREF
         TM(I,J,K,IZ)=TMIN
         DEN(I,J,K,IZ)=DNIN
         DK(I,J,K,IZ)=DKIN
         DE(I,J,K,IZ)=DEIN
         IF(I .EQ. 1 .OR. J .EQ. JZT(IZ)) THEN
            U(I,J,K,IZ)=0.
            DK(I,J,K,IZ)=0.
            DE(I,J,K,IZ)=0.
         ENDIF
         IF(I .EQ. IZT(IZ) .AND. J .GE. JC2) THEN
            U(I,J,K,IZ)=0.
            DK(I,J,K,IZ)=0.
            DE(I,J,K,IZ)=0.
         ENDIF
      ENDDO
   ENDDO
6100   CONTINUE
6000 CONTINUE

95
DO KK=1,NGAS
   FM(I,J,K,KK,IZ)=FMIN(KK)
ENDDO
6100 CONTINUE
6000 CONTINUE

C-------- nozzle flow initialization
IZ=2
I=1
IT1=IZT(1)
DO J=1,JZT(IZ)
   DO K=1,KZT(IZ)
      U(I,J,K,IZ)=U(IT1,J,K,1)
      V(I,J,K,IZ)=V(IT1,J,K,1)
      W(I,J,K,IZ)=W(IT1,J,K,1)
      P(I,J,K,IZ)=P(IT1,J,K,1)
      TM(I,J,K,IZ)=TM(IT1,J,K,1)
      DEN(I,J,K,IZ)=DEN(IT1,J,K,1)
      DK(I,J,K,IZ)=DK(IT1,J,K,1)
      DE(I,J,K,IZ)=DE(IT1,J,K,1)
      IF (J .EQ. JZT(IZ)) THEN
         U(I,J,K,IZ)=0.
         V(I,J,K,IZ)=0.
         W(I,J,K,IZ)=0.
         DK(I,J,K,IZ)=0.
         DE(I,J,K,IZ)=0.
      ENDIF
   DO KK=1,NGAS
      FM(I,J,K,KK,IZ)=FM(IT1,J,K,KK,1)
   ENDDO
   ENDDO
   ENDDO
JT=JZT(IZ)
AREAT=Y(IN2,JT,1,IZ)**2
AREA0=Y(1,JT,1,IZ)**2
FLOW0=DNIN*UIN*AREA0
ARAT=AMIN1(1.,AREAT/AREA0)
CALL NOZDAT(1,ARAT,DNRAT,TMRAT)
DN0=DNIN/DNRAT
TM0=TMIN/TMRAT
WRITE(6,4002)
4001 FORMAT(5X,I3,2X,3(F15.8,2X))
4002 FORMAT(6X,'I',7X,'DENSITY',8X,'TEMPERATURE',8X,'PRESSURE',/5X,
     &'===',2X,3(15H============,2X))
DO I=2,IN2
   AREAP=Y(I,JT,1,IZ)**2
   ARAT=AMIN1(1.,AREAT/AREAP)
   CALL NOZDAT(1,ARAT,DNRAT,TMRAT)
   DNP=DN0*DNRAT
   TMP=TM0*TMRAT
   ULOCAL=FLOW0/(AREAP*DNP)
   PP=DNP*TMP*PREF/DNIN/TMIN
   WRITE(6,4001) I,DNP,TMP,PP
DO J=1,JT
   DX=X(I,J,1,IZ)-X(I-1,J,1,IZ)
   DR=Y(I,J,1,IZ)-Y(I-1,J,1,IZ)
   ANG=ATAN2(DR,DX)
   IF (I .EQ. IN2) ANG=0.
   DO K=1,KZT(IZ)
U(I,J,K,IZ)=ULOCAL
V(I,J,K,IZ)=ULOCAL*TAN(ANG)
W(I,J,K,IZ)=0.
P(I,J,K,IZ)=PP
DEN(I,J,K,IZ)=DNP
TM(I,J,K,IZ)=TMP
DK(I,J,K,IZ)=DK(1,J,K,IZ)
DE(I,J,K,IZ)=DE(1,J,K,IZ)
IF(J .EQ. JT) THEN
  U(I,J,K,IZ)=0.
  V(I,J,K,IZ)=0.
  DK(I,J,K,IZ)=0.
  DE(I,J,K,IZ)=0.
ENDIF
DO KK=1,NGAS
  FM(I,J,K,KK,IZ)=FM(1,J,K,KK,IZ)
ENDDO
ENDDO
ENDDO
ENDDO
DO I=IN2+1,IZT(IZ)
  AREAP=Y(I,JT,1,IZ)**2
  ARAT=AMIN1(1.,AREAT/AREAP)
  CALL NOZDAT(2,ARAT,DNRAT,TMRAT)
  DNP=DN0*DNRAT
  TMP=TM0*TMRAT
  ULOCAL=FLOW0/(AREAP*DNP)
  PP=DNP*TMP*PREF/DNIN/TMIN
  WRITE(6,4001) I,DNP,TMP,PP
  DO J=1,JT
    DX=X(I,J,1,IZ)-X(I-1,J,1,IZ)
    DR=Y(I,J,1,IZ)-Y(I-1,J,1,IZ)
    ANG=ATAN2(DR,DX)
    DO K=1,KZT(IZ)
      U(I,J,K,IZ)=ULOCAL
      V(I,J,K,IZ)=ULOCAL*TAN(ANG)
      W(I,J,K,IZ)=0.
      P(I,J,K,IZ)=PP
      DEN(I,J,K,IZ)=DNP
      TM(I,J,K,IZ)=TMP
      DK(I,J,K,IZ)=DK(1,J,K,IZ)
      DE(I,J,K,IZ)=DE(1,J,K,IZ)
      IF(J .EQ. JT) THEN
        U(I,J,K,IZ)=0.
        V(I,J,K,IZ)=0.
        DK(I,J,K,IZ)=0.
        DE(I,J,K,IZ)=0.
      ENDIF
      DO KK=1,NGAS
        FM(I,J,K,KK,IZ)=FM(1,J,K,KK,IZ)
      ENDDO
    ENDDO
  ENDDO
ENDDO
ENDDO
ENDDO
ENDDO
DO I=IN2+1,IZT(IZ)
  AREAP=Y(I,JT,1,IZ)**2
  ARAT=AMIN1(1.,AREAT/AREAP)
  CALL NOZDAT(2,ARAT,DNRAT,TMRAT)
  DNP=DN0*DNRAT
  TMP=TM0*TMRAT
  ULOCAL=FLOW0/(AREAP*DNP)
  PP=DNP*TMP*PREF/DNIN/TMIN
  WRITE(6,4001) I,DNP,TMP,PP
  DO J=1,JT
    DX=X(I,J,1,IZ)-X(I-1,J,1,IZ)
    DR=Y(I,J,1,IZ)-Y(I-1,J,1,IZ)
    ANG=ATAN2(DR,DX)
    DO K=1,KZT(IZ)
      U(I,J,K,IZ)=ULOCAL
      V(I,J,K,IZ)=ULOCAL*TAN(ANG)
      W(I,J,K,IZ)=0.
      P(I,J,K,IZ)=PP
      DEN(I,J,K,IZ)=DNP
      TM(I,J,K,IZ)=TMP
      DK(I,J,K,IZ)=DK(1,J,K,IZ)
      DE(I,J,K,IZ)=DE(1,J,K,IZ)
      IF(J .EQ. JT) THEN
        U(I,J,K,IZ)=0.
        V(I,J,K,IZ)=0.
        DK(I,J,K,IZ)=0.
        DE(I,J,K,IZ)=0.
      ENDIF
      DO KK=1,NGAS
        FM(I,J,K,KK,IZ)=FM(1,J,K,KK,IZ)
      ENDDO
    ENDDO
  ENDDO
ENDDO
ENDDO
ENDDO
ENDDO
C
C----- Injector inlet conditions
C
IZ=1
J=J2T(I2)
DO I=IH5,IH6
DO K=KC2,KC4
IF(I.NE.IH5 .AND. I.NE.IH6 .AND. K.NE.KC2 .AND. K.NE.KC4) THEN
  W(I,J,K,I2)=UOX*COS(TCHAMB)
  U(I,J,K,I2)=-UOX*TAN(AOXINJ)
  V(I,J,K,I2)=-UOX*SIN(TCHAMB)
  DK(I,J,K,I2)=DKOX
  DE(I,J,K,I2)=DEOX
ENDIF
TM(I,J,K,I2)=TOX
DEN(I,J,K,I2)=DNOX
DO KK=1,NGAS
  FM(I,J,K,KK,I2)=FMOX(KK)
ENDDO
ENDDO
ENDDO
DO I=ID5,ID6
DO K=KC6,KC7
  W(I,J,K,I2)=UFG*COS(2.*TCHAMB)
  U(I,J,K,I2)=-UFG*TAN(ARPINJ)
  V(I,J,K,I2)=-UFG*SIN(2.*TCHAMB)
  DK(I,J,K,I2)=DKFG
  DE(I,J,K,I2)=DEFG
ENDIF
TM(I,J,K,I2)=TFG
DEN(I,J,K,I2)=DNFG
DO KK=1,NGAS
  FM(I,J,K,KK,I2)=FMFG(KK)
ENDDO
ENDDO
ENDDO
K=KC1
DO I=ID5,ID6
  TM(I,J,K,I2)=TFG
  DEN(I,J,K,I2)=DNFG
  DO KK=1,NGAS
    FM(I,J,K,KK,I2)=FMFG(KK)
  ENDDO
ENDDO
ENDDO
DO I=IH3,IH4
DO K=KC3,KC4
IF(I.NE.IH3 .AND. I.NE.IH4 .AND. K.NE.KC3 .AND. K.NE.KC4) THEN
  W(I,J,K,I2)=UFG*COS(TCHAMB)
  U(I,J,K,I2)=UFG*TAN(ARPINJ)
  V(I,J,K,I2)=-UFG*SIN(TCHAMB)
  DK(I,J,K,I2)=DKFG
  DE(I,J,K,I2)=DEFG
ENDIF
TM(I,J,K,I2)=TFG
DEN(I,J,K,I2)=DNFG
DO KK=1,NGAS
  FM(I,J,K,KK,I2)=FMFG(KK)
ENDDO
ENDDO
ENDDO
ENDDO
DO I=ID3,ID4

DO K=KC5,KC7
   IF(I.NE.ID3 .AND. I.NE.ID4 .AND. K.NE.KC5 .AND. K.NE.KC7) THEN
      W(I,J,K,IZ)=UOX*COS(2.*TCHAMB)
      U(I,J,K,IZ)=UOX*TAN(AOXINJ)
      V(I,J,K,IZ)=-UOX*SIN(2.*TCHAMB)
      DK(I,J,K,IZ)=DKOX
      DE(I,J,K,IZ)=DEOX
   ENDIF
   TM(I,J,K,IZ)=TOX
   DEN(I,J,K,IZ)=DNOX
   DO KK=1,NGAS
      FM(I,J,K,KK,IZ)=FMOX(KK)
   ENDDO
ENDDO
ENDDO
DO I=ID3,ID4
   TM(I,J,K,IZ)=TOX
   DEN(I,J,K,IZ)=DNOX
   DO KK=1,NGAS
      FM(I,J,K,KK,IZ)=FMOX(KK)
   ENDDO
ENDDO

C
C------- PRINT OUT THE PRE-PROCESS GRID & FLOW FILES
INSO1=1
INSO4=1
INSOKE=1
INSO7=2
WRITE(IW1) IZON
DO 9000 IZ=1,IZON
   WRITE(IW1) IZT(IZ),JZT(IZ),KZT(IZ)
9000 CONTINUE
DO 9100 IZ=1,IZON
   IS=IZS(IZ)
   IT=IZT(IZ)
   JS=JZS(IZ)
   JT=JZT(IZ)
   KS=KZS(IZ)
   KT=KZT(IZ)
   WRITE(IW1)(((X(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
   WRITE(IW1)(((Y(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
   WRITE(IW1)(((Z(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
9100 CONTINUE
WRITE(IW2) INSO1,INSO4,INSOKE,INSO7,NGAS
DO 9200 IZ=1,IZON
   IS=IZS(IZ)
   IT=IZT(IZ)
   JS=JZS(IZ)
   JT=JZT(IZ)
   KS=KZS(IZ)
   KT=KZT(IZ)
   WRITE(IW2)(((DEN(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
   WRITE(IW2)(((U(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
   WRITE(IW2)(((V(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
   WRITE(IW2)(((W(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
   WRITE(IW2)(((P(I,J,K,IZ),I=IS,IT),J=JS,JT),K=KS,KT)
9200 CONTINUE

99
WRITE(IW2)(((TM(I,J,K,I2),I=IS,IT),J=JS,JT),K=KS,KT)
WRITE(IW2)(((DK(I,J,K,I2),I=IS,IT),J=JS,JT),K=KS,KT)
WRITE(IW2)(((DE(I,J,K,I2),I=IS,IT),J=JS,JT),K=KS,KT)
WRITE(IW2)(((TM(I,J,K,I2),I=IS,IT),J=JS,JT),K=KS,KT)
WRITE(IW2)(((TM(I,J,K,I2),I=IS,IT),J=JS,JT),K=KS,KT)
IF(NGAS .GT. 0) THEN
   DO KK=1,NGAS
      WRITE(IW2)(((FM(I,J,K,KK,I2),I=IS,IT),J=JS,JT),K=KS,KT)
   ENDDO
ENDIF
9200 CONTINUE
1 FORMAT(15I5)
2 FORMAT(5(1P,E16.8))
3 FORMAT(8I5)
STOP
END
C
C     ****************************************************************
C     SUBROUTINE SPCNTL IS USED TO PACK THE GRID IN VARIOUS DIRECTIONS
C     ****************************************************************
C
SUBROUTINE SPCNTL(I,I1,I2,ITYPE,DTA,FACT)
C     ========================================================
C       ITYPE < 0: Power law packing.
C       ITYPE = 0: Uniform spacing.
C       ITYPE > 0: Hyper Tangent packing
C     ========================================================
IF(ITYPE .EQ. 0) THEN
   FACT=FLOAT(I-I1)/FLOAT(I2-I1)
ELSE IF(ITYPE .EQ. 1) THEN
   RAT=1.-FLOAT(I-I1)/FLOAT(I2-I1)
   FACT=1.-TANH(DTA*RAT)/TANH(DTA)
ELSE IF(ITYPE .EQ. 2) THEN
   RAT=FLOAT(I-I1)/FLOAT(I2-I1)
   FACT=TANH(DTA*RAT)/TANH(DTA)
ELSE IF(ITYPE .EQ. 3) THEN
   RAT=2.*FLOAT(I-I1)/FLOAT(I2-I1)-1.
   FACT=0.5*(1.+TANH(DTA*RAT)/TANH(DTA))
ELSE
   WRITE(6,99) ITYPE
ENDIF
99  FORMAT('??? ERROR IN SPECIFYING TYPE OF GRID PACKING, ITYPE = ',
&I4)
RETURN
END
C
C     *******************************************************
C     *                                                        *
C     *  SUBROUTINE TRANSF EMPLOY TRANSFINITE INTERPOLATION TO   *
C     *  GENERATE ALGEBRAIC GRIDS BASED ON THE PRESCRIBED BOUNDARY *
C     *  GRID POINTS                                            *
C     *******************************************************
SUBROUTINE TRANSF(I1, I2, J1, J2, XX, ZZ)
DIMENSION XX(100, 100), ZZ(100, 100)
DIMENSION X1(100, 100), X2(100, 100), Z1(100, 100), Z2(100, 100)
DO 110 I = I1, I2
  DO 110 J = J1, J2
    PXI1 = FLOAT(I2 - I) / FLOAT(I2 - I1)
    PXI2 = FLOAT(I - I1) / FLOAT(I2 - I1)
    X1(I, J) = PXI1 * XX(I1, J) + PXI2 * XX(I2, J)
    Z1(I, J) = PXI1 * ZZ(I1, J) + PXI2 * ZZ(I2, J)
  110 CONTINUE
DO 120 I = I1, I2
  DO 120 J = J1, J2
    PETA1 = FLOAT(J2 - J) / FLOAT(J2 - J1)
    PETA2 = FLOAT(J - J1) / FLOAT(J2 - J1)
    X2(I, J) = PETA1 * (XX(I, J1) - X1(I, J1)) + PETA2 * (XX(I, J2)
      & - X1(I, J2))
    Z2(I, J) = PETA1 * (ZZ(I, J1) - Z1(I, J1)) + PETA2 * (ZZ(I, J2)
      & - Z1(I, J2))
  120 CONTINUE
DO 130 I = I1, I2
  DO 130 J = J1, J2
    XX(I, J) = X1(I, J) + X2(I, J)
    ZZ(I, J) = Z1(I, J) + Z2(I, J)
  130 CONTINUE
RETURN
END

SUBROUTINE NOZDAT IS USED TO INTERPOLATE NOZZLE FLOW FIELD BASED
ON 1-D ISENTROPIC PROFILE

SUBROUTINE NOZDAT(ISEC, ARAT, DN RAT, TM RAT)

C-----1-D NOZZLE DATA
  DIMENSION AR1(11), DN1(11), TP1(11), AR2(11), DN2(11), TP2(11)
C-----SUBSONIC
  DATA AR1/0.00000, 0.10350, 0.20560, 0.30510, 0.40070, 0.50590,
    & 0.60290, 0.70190, 0.80630, 0.90120, 1.00000/
  DATA DN1/1.00000, 0.99820, 0.99280, 0.98400, 0.97180, 0.95350,
    & 0.93130, 0.90160, 0.85890, 0.80160, 0.63390/
  DATA TP1/1.00000, 0.99930, 0.99710, 0.99360, 0.98860, 0.98110,
    & 0.97190, 0.95940, 0.94100, 0.91530, 0.83330/
C-----SUPERSONIC
  DATA AR2/0.00000, 0.10210, 0.20090, 0.30250, 0.40100, 0.50320,
    & 0.60290, 0.70020, 0.80500, 0.90130, 1.00000/
  DATA DN2/0.00000, 0.03044, 0.06363, 0.10100, 0.14080, 0.18630,
    & 0.23520, 0.29000, 0.35950, 0.44180, 0.63390/
  DATA TP2/0.00000, 0.24740, 0.33230, 0.39960, 0.45670, 0.51040,
    & 0.56050, 0.60950, 0.66420, 0.72130, 0.83330/
IDPT = 11
C-END-1-D NOZZLE DATA
ITEST=0
ARAT=AMIN1(1.,ARAT)
DNRAT=0.
TMRAT=0.
IF(ISEC .EQ. 1) THEN
  DO 1000 II=2,IDPT
    IF(ITEST .GE. 1) GO TO 1100
    P1=ARAT-AR1(II-1)
    P2=AR1(II)-ARAT
    IF(P1*P2 .GE. 0.) THEN
      DNRAT=(P1*DN1(II)+P2*DN1(II-1))/(P1+P2)
      TMRAT=(P1*TP1(II)+P2*TP1(II-1))/(P1+P2)
      ITEST=1
    END IF
  1000 CONTINUE
1100 CONTINUE
ELSE IF(ISEC .EQ. 2) THEN
  DO 2000 II=2,IDPT
    IF(ITEST .GE. 1) GO TO 2100
    P1=ARAT-AR2(II-1)
    P2=AR2(II)-ARAT
    IF(P1*P2 .GE. 0.) THEN
      DNRAT=(P1*DN2(II)+P2*DN2(II-1))/(P1+P2)
      TMRAT=(P1*TP2(II)+P2*TP2(II-1))/(P1+P2)
      ITEST=1
    END IF
  2000 CONTINUE
2100 CONTINUE
END IF
RETURN
END
C
C  **************************************************************
C  *                                                            *
C  *  SUBROUTINE TRAN3D EMPLOY 3-D TRANSFINITE INTERPOLATION TO  *
C  *  GENERATE ALGEBRAIC GRIDS BASED ON THE PRESCRIBED BOUNDARY  *
C  *  GRID POINTS                                               *
C  *                                                            *
C  **************************************************************
C
SUBROUTINE TRAN3D(I1,I2,J1,J2,K1,K2,XX,YY,ZZ)
PARAMETER (NI = 51, NJ = 51, NK=61)
DIMENSION XX(NI,NJ,NK),YY(NI,NJ,NK),ZZ(NI,NJ,NK)
DIMENSION X1(NI,NJ,NK),X2(NI,NJ,NK),X3(NI,NJ,NK),
  &  Y1(NI,NJ,NK),Y2(NI,NJ,NK),Y3(NI,NJ,NK),
  &  Z1(NI,NJ,NK),Z2(NI,NJ,NK),Z3(NI,NJ,NK)
C
DO 110 I=I1,I2
DO 110 J=J1,J2
DO 110 K=K1,K2
  PXI1=FLOAT(I2-I)/FLOAT(I2-I1)
  PXI2=FLOAT(I-I1)/FLOAT(I2-I1)
  X1(I,J,K)=PXI1*XX(I,J,K)+PXI2*XX(I2,J,K)
  Y1(I,J,K)=PXI1*YY(I,J,K)+PXI2*YY(I2,J,K)
  Z1(I,J,K)=PXI1*ZZ(I,J,K)+PXI2*ZZ(I2,J,K)
110 CONTINUE
DO 120 I=I1,I2
DO 120 J=J1,J2
DO 120 K=K1,K2
   PETA1=FLOAT(J2-J)/FLOAT(J2-J1)
PETA2=FLOAT(J-J1)/FLOAT(J2-J1)
   X2(I,J,K)=PETA1* (XX(I,J1,K)-X1(I,J1,K))
   & +PETA2* (XX(I,J2,K)-X1(I,J2,K))
   Y2(I,J,K)=PETA1* (YY(I,J1,K)-Y1(I,J1,K))
   & +PETA2* (YY(I,J2,K)-Y1(I,J2,K))
   Z2(I,J,K)=PETA1* (ZZ(I,J1,K)-Z1(I,J1,K))
   & +PETA2* (ZZ(I,J2,K)-Z1(I,J2,K))
120  CONTINUE

DO 130 I=I1,I2
   DO 130 J=J1,J2
   DO 130 K=K1,K2
      PZTA1=FLOAT(K2-K)/FLOAT(K2-K1)
PZTA2=FLOAT(K-K1)/FLOAT(K2-K1)
      X3(I,J,K)=PZTA1* (XX(I,J,K1)-X1(I,J,K1)-X2(I,J,K1))
      Y3(I,J,K)=PZTA1* (YY(I,J,K1)-Y1(I,J,K1)-Y2(I,J,K1))
      Z3(I,J,K)=PZTA1* (ZZ(I,J,K1)-Z1(I,J,K1)-Z2(I,J,K1))
130  CONTINUE

DO 140 I=I1,I2
   DO 140 J=J1,J2
   DO 140 K=K1,K2
      XX(I,J,K)=X1(I,J,K)+X2(I,J,K)+X3(I,J,K)
      YY(I,J,K)=Y1(I,J,K)+Y2(I,J,K)+Y3(I,J,K)
      ZZ(I,J,K)=Z1(I,J,K)+Z2(I,J,K)+Z3(I,J,K)
140  CONTINUE
RETURN
END

C SUBROUTINE IE4(I1,I2,I3,I4,J1,J2,J3,J4)
C
I1=J1
I2=J2
I3=J3
I4=J4
RETURN
END
APPENDIX D

THE VORTEX ENGINE SIMULATION

The sample test presented to illustrate several unique features of the FDNS-RFV/PVM code is a numerical simulation of a vortex engine with LOX and RP-1 as the propellants. The concept of the vortex engine is to use the vorticity of the injected propellants to provide longer residence times in the chamber and, hopefully, better mixing. A preliminary design of such a vortex motor was analyzed. This motor introduced the propellants through 16 pairs of unlike injector elements, tangential to and around the circumference of the combustion chamber. The injected propellants therefore produced swirl in the chamber to enhance the residence time in the short chamber. The propellants are injected at a high enough velocity to create the swirl, and they do mix very well at the injection point. In a conventional motor, the many injector elements provide well mixed propellants, whereas the vortex motor depends on passive mixing of the propellant streams. In the sample test case, the injection sequence of the propellants is alternating. Hence, a one-eighth pie-shaped section of the chamber was simulated. Periodic boundary conditions were used on the sides of the section. The simulation was conducted only part way down the expansion nozzle, but this was far enough to evaluate the mixing efficiency in the chamber. Operating conditions for this engine are expected to be a mixture ratio of 2.6, 4.15 lbm/sec total propellant flow, 1000 psia chamber pressure. The chamber diameter is 2.35 in, throat diameter 0.96 in, and exit diameter of 3.03 in for an expansion ratio of 9.96. The diameters of the RP-1 injector and the LOX injector are 0.04 and 0.06 in, respectively. The impingement angle is 36° between the unlike injector elements. The computational domain consists of 180K grid points.

In order to simulate the liquid spray of LOX and RP-1, the homogeneous real-fluid model is employed to treat the propellant as continuum because the chamber pressure is well above the critical pressure of both LOX and RP-1. In addition, the thermal properties of RP-1 fuel are required for simulating the spray combustion. However, the RP-1 fuel consists of a complex mixture of many components and their exact composition vary from batch to batch. Core Laboratory measurements of RP-1 indicated the constituents to be about 42% liquid volume of paraffins, 25% monocycloparaffins, 24% di- and tri-cycloparaffins, 3% benzenes and 6% naphthalenes. Examination of the constituents revealed about 12% liquid volume of molecules with 10 carbon atoms (C10's), 13% C11's, 29% C12's, 25% C13's, 11% C14's and the rest either larger or smaller. In order to simplify the description of RP-1, a surrogate fuel model was used. The carbon breakdown was reduced to 20% C11's, 30% C12's, 30% C13's and 20% C14's. Eighteen surrogate constituents were identified which represented the Core Laboratory data. The surrogate properties compared well to a RP-1 molecular weight of 175, hydrogen to carbon ratio (H/C) of 1.95 and liquid density of 0.83 g/cc. A reasonable value of H/C was unattainable due to the low H/C's for aromatics. Since the reported percentage (9%) of aromatics was higher than reported by other sources, and since complex dinuclear aromatics (naphthalenes) are often indistinguishable from polycyclic paraffins (naphthenes), the naphthalenes percentage was reduced by 2% and the polycyclic paraffin percentage was increased by 2%. As a result, a one-formula surrogate fuel for RP-1 was modeled as $C_{12.39}H_{24.15}$, which has an H/C close to 1.95 and a molecular weight of 173.15. The true critical properties for the RP-1 surrogate fuel were calculated using Gibbs method (all prediction methods cited were taken from Reid, et al\textsuperscript{D-1}). Unknown saturated liquid volumes of the constituents were estimated using the method of Gunn and Yamada. Unknown critical properties were estimated using the group contribution method of Joback. These methods were utilized to obtain the fluid and
thermal properties of the RP-1 surrogate fuel. Moreover, it is well known that various light hydrocarbon species present under the fuel rich combustion condition. However, for practical engineering application it is not feasible and efficient to include all hydrocarbon species and their corresponding chemical reactions. A light hydrocarbon species, C$_2$H$_4$ surrogate which has the same heat capacity as C$_2$H$_4$ and an identical heat of formation as RP-1, is employed to approximate the lump sum of all small hydrocarbon species. Since the H/C ratio of C$_2$H$_4$ surrogate is 2 which is slightly different from 1.95, a simplification step was taken to model RP-1 as C$_{12.4}$H$_{24.8}$ surrogate. The heat of formation was also modified to reflect the change of H/C ratio. The combustion is represented with four global kinetics expressions, as shown in Table D-1, to simulate RP-1 pyrolysis, oxidation of pyrolysis gas, soot formation, and soot oxidation process, where the RP-1 oxidation step produces H$_2$ and CO. The combustion was completed with either a wet CO finite-rate model.

References:

**Table D-1**

<table>
<thead>
<tr>
<th>QUASIGLOBAL KINETICS MODEL WITH SOOT FORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Mechanism</td>
</tr>
<tr>
<td>RP-1 Pyrolysis</td>
</tr>
<tr>
<td>( \text{C}<em>{12.4}\text{H}</em>{24.8} ) ( \rightarrow ) ( \text{C}_2\text{H}_4 )</td>
</tr>
<tr>
<td>Oxidation of Pyrolysis Gas</td>
</tr>
<tr>
<td>( \text{C}_2\text{H}_4 + \text{O}_2 \rightarrow \text{CO} + \text{H}_2 )</td>
</tr>
<tr>
<td>Soot Formation</td>
</tr>
<tr>
<td>( \text{C}_2\text{H}_4 \rightarrow \text{C}<em>9\text{H}</em>{24} )</td>
</tr>
</tbody>
</table>

Note: \([\dot{C}_i]= - A T^B \exp \{-E/RT\} [C_i]^A [C_j]^B\)  
\([C_i]\) concentrations (gm moles/cm³)

Units on A are such that units on \([\dot{C}_i]\) are correct

**SOOT OXIDATION MODEL**

\[
\text{Rate} = \frac{72 M_w R_u T}{\rho_s D_s} \left[ \frac{K_1 \psi}{1 + K_4 P_{O_2}} + K_2 (1 - \psi) \right] [\text{Soot}][\text{O}_2]
\]

where \(\psi = \left(1 + \frac{P_{O_2} K_3}{K_2}\right)^{-1}\); \(K_i = A_i \exp\left(-\frac{E_i}{R T}\right); i = 1 \rightarrow 4\)

\(A_1 = 20, E_1/R = 1.509 \times 10^4\)  
\(A_2 = 4.46 \times 10^3, E_2/R = 7.6497 \times 10^3\)  
\(A_3 = 1.51 \times 10^5, E_3/R = 4.8817 \times 10^4\)  
\(A_4 = 21.3, E_4/R = -2.0634 \times 10^3\)  
\(R_u = 82.06 \text{ atm} @ \text{cm}^3/\text{g-mole} @ \text{°K}\)  
\(\rho_s = 1.86 \text{ g/cm}^3\)  
\(D_s = 40 \text{ nm (assumed value)}\)  
\(M_w = 1177.25 \text{ g/g-mole}\)  
\(P_{O_2} = \text{partial pressure of O}_2, \text{ in atm}\)

Note: Soot is defined as \(\text{C}_9\text{H}_{24}\) in order to treat it as an idea gas. This approximation makes incipient soot particles the right size and maintains the right carbon-to-hydrogen ratio for soot.