NASA CONTRACTOR
REPORT

NASA CR-178818

A COMPUTER CODE FOR THREE-DIMENSIONAL INCOMPRESSIBLE
FLOWS USING NONORTHOGONAL BODY-FITTED COORDINATE SYSTEMS

By Y. S. Chen
Universities Space Research Association
Science and Engineering Directorate
Atmospheric Sciences Division
Systems Dynamics Laboratory

Final Report

March 1986

Prepared for
NASA-Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812
In this report, a numerical method for solving the equations of motion of three-dimensional incompressible flows in nonorthogonal body-fitted coordinate (BFC) systems has been developed. The equations of motion are transformed to a generalized curvilinear coordinate system from which the transformed equations are discretized using finite difference approximations in the transformed domain. The hybrid scheme is used to approximate the convection terms in the governing equations. Solutions of the finite difference equations are obtained iteratively by using a pressure-velocity correction algorithm (SIMPLE-C). Numerical examples of two- and three-dimensional, laminar and turbulent flow problems are employed to evaluate the accuracy and efficiency of the present computer code. The user's guide and computer program listing of the present code are also included.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>TRANSFORMATION OF THE EQUATIONS OF MOTION</td>
<td>2</td>
</tr>
<tr>
<td>DISCRETIZATION OF THE EQUATIONS OF MOTION</td>
<td>5</td>
</tr>
<tr>
<td>SOLUTION PROCEDURES</td>
<td>7</td>
</tr>
<tr>
<td>NUMERICAL EXAMPLES</td>
<td>12</td>
</tr>
<tr>
<td>A. 2-D Laminar Driven Square-Cavity Flows</td>
<td>12</td>
</tr>
<tr>
<td>B. 2-D Laminar Flows Over a Backward-Facing Step</td>
<td>15</td>
</tr>
<tr>
<td>C. 2-D Turbulent Flows Over a Backward-Facing Step</td>
<td>18</td>
</tr>
<tr>
<td>D. Developing Laminar Flow Inside a 90-Deg-Bend Square Duct</td>
<td>18</td>
</tr>
<tr>
<td>CONCLUSIONS</td>
<td>23</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>24</td>
</tr>
<tr>
<td>APPENDIX A. COMPUTER CODE STRUCTURES AND USER'S GUIDE</td>
<td>27</td>
</tr>
<tr>
<td>APPENDIX B. LIST OF FORTRAN SYMBOLS</td>
<td>31</td>
</tr>
<tr>
<td>APPENDIX C. PROGRAM LISTING</td>
<td>41</td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>1.</td>
<td>Three-dimensional grid structure and labeling around a grid node P</td>
</tr>
<tr>
<td>2.</td>
<td>Locations where the variables are stored</td>
</tr>
<tr>
<td>3.</td>
<td>Control volumes where the mass conservation is evaluated for solving the pressure correction equation</td>
</tr>
<tr>
<td>4.</td>
<td>Physical geometry and wall boundary conditions for laminar flows inside a wall-driven square cavity</td>
</tr>
<tr>
<td>5.</td>
<td>Mesh systems used for driven cavity problem</td>
</tr>
<tr>
<td>6.</td>
<td>Velocity vector plots</td>
</tr>
<tr>
<td>7.</td>
<td>Comparisons of velocity profiles along the mid-section of the square cavity</td>
</tr>
<tr>
<td>8.</td>
<td>Convergence history for the driven cavity problem, Re = 400</td>
</tr>
<tr>
<td>9.</td>
<td>Physical geometry and boundary conditions of laminar flows over a backward-facing step</td>
</tr>
<tr>
<td>10.</td>
<td>Reattachment length versus Reynolds number for laminar flows over a backward-facing step</td>
</tr>
<tr>
<td>11.</td>
<td>Streamline plots for laminar flow over a backward-facing step</td>
</tr>
<tr>
<td>12.</td>
<td>Physical geometry and boundary conditions of turbulent flows over a backward-facing step</td>
</tr>
<tr>
<td>13.</td>
<td>Locus of flow reversal inside the recirculation region for turbulent flow over a backward-facing step</td>
</tr>
<tr>
<td>14.</td>
<td>Streamline pattern of turbulent flow over a backward-facing step with 2:3 expansion ratio</td>
</tr>
<tr>
<td>15.</td>
<td>Contours of turbulent kinetic energy of turbulent flow over a backward-facing step with 2:3 expansion ratio</td>
</tr>
<tr>
<td>16.</td>
<td>Geometry and mesh system of a 90-deg-bend square duct developing laminar flow problem</td>
</tr>
<tr>
<td>17.</td>
<td>Primary velocity patterns of laminar flow inside a 90-deg-bend square duct</td>
</tr>
<tr>
<td>18.</td>
<td>Secondary velocity patterns of laminar flow inside a 90-deg-bend</td>
</tr>
<tr>
<td>19.</td>
<td>Primary velocity profiles for a 3-D 90-deg-bend square duct</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>A</td>
<td>link coefficient of finite difference equation</td>
</tr>
<tr>
<td>A_p</td>
<td>link coefficient of the node at the center of a control volume</td>
</tr>
<tr>
<td>A^u</td>
<td>link coefficient for the u-equation</td>
</tr>
<tr>
<td>A^v</td>
<td>link coefficient for the v-equation</td>
</tr>
<tr>
<td>A^w</td>
<td>link coefficient for the w-equation</td>
</tr>
<tr>
<td>A^o</td>
<td>link coefficient for time marching scheme</td>
</tr>
<tr>
<td>C_v</td>
<td>specific heat constant at constant volume</td>
</tr>
<tr>
<td>C_1</td>
<td>turbulence model constant, $= 1.44$</td>
</tr>
<tr>
<td>C_2</td>
<td>turbulence model constant, $= 1.92$</td>
</tr>
<tr>
<td>C_μ</td>
<td>turbulence model constant, $= 0.09$</td>
</tr>
<tr>
<td>D</td>
<td>diffusion coefficient for the pressure correction equation</td>
</tr>
<tr>
<td>e</td>
<td>internal energy per unit mass (jou/m^3)</td>
</tr>
<tr>
<td>J</td>
<td>Jacobian of the metric transformation</td>
</tr>
<tr>
<td>k</td>
<td>turbulence kinetic energy (m^3/s^2)</td>
</tr>
<tr>
<td>k'</td>
<td>thermal conductivity of the fluid</td>
</tr>
<tr>
<td>P</td>
<td>pressure in the fluid (N/m^2)</td>
</tr>
<tr>
<td>P_r</td>
<td>production term for the turbulent kinetic energy</td>
</tr>
<tr>
<td>Q</td>
<td>energy added per unit volume (jou/m^3)</td>
</tr>
<tr>
<td>S</td>
<td>source term</td>
</tr>
<tr>
<td>S_u</td>
<td>source term of the u-equation</td>
</tr>
<tr>
<td>S_v</td>
<td>source term of the v-equation</td>
</tr>
<tr>
<td>S_w</td>
<td>source term of the w-equation</td>
</tr>
<tr>
<td>T</td>
<td>temperature (°K)</td>
</tr>
<tr>
<td>t</td>
<td>time (sec)</td>
</tr>
<tr>
<td>u</td>
<td>velocity in x direction</td>
</tr>
</tbody>
</table>
v velocity in y direction
w velocity in z direction
X X-coordinate (m)
Y Y-coordinate (m)
Z Z-coordinate (m)

Greek
\[ \Gamma \] diffusion coefficient
\[ \varepsilon \] turbulent kinetic energy dissipation rate \( (m^2/s^3) \)
\[ \Delta \] difference operator
\[ \phi \] variable of general transport equation
\[ \phi^0 \] solution at the previous time level
\[ \sigma_k \] turbulence model constant, = 1.0
\[ \sigma_\varepsilon \] turbulence model constant, = 1.3
\[ \xi \] curvilinear coordinate
\[ \zeta \] curvilinear coordinate
\[ \mu \] molecular viscosity \( (N-S/m^2) \)
\[ \mu_t \] turbulent eddy viscosity \( (N-S/m^2) \)
\[ \mu_{\text{eff}} \] effective viscosity \( (N-S/m^2) \)
\[ \rho \] density \( (kg/m^3) \)
\[ \Sigma_{i} \] summation over all values around a grid node P
\[ \eta \] curvilinear coordinate

Subscript
\[ i \] index of all possible values
ref reference value
\[ \text{max} \] maximum quantity
Superscript

o previous time level solution

* current solution

' correction quantity
CONTRACTOR REPORT

A COMPUTER CODE FOR THREE-DIMENSIONAL INCOMPRESSIBLE FLOWS USING NONORTHOGONAL BODY-FITTED COORDINATE SYSTEMS

INTRODUCTION

With the currently increasing computer capability and various flow solvers developed, numerical simulations of three-dimensional incompressible flow problems using Reynolds-average Navier-Stokes equations are now becoming more feasible in many engineering design and analysis applications. In many real world flow problems, the boundary geometries are complex such that it is more accurate to describe the geometries using body-fitted coordinate (BFC) systems. Especially for internal flow problems with complex geometries such as those of the hot gas manifold (HGM) of the Space Shuttle Main Engine (SSME), the use of nonorthogonal BFC systems for numerical solutions can be beneficial in many aspects. It is not only the boundary geometries that can be represented more closely using BFC systems, but also grid-refined solutions can be obtained without increasing an excessive amount in computer memory. In addition, once a particular flow problem has been set up, the redesign or optimization process of the boundary shapes can be performed very easily using BFC systems.

Several numerical methods [1, 2, 3, 4, 5, 6] has been developed for solving the incompressible Navier-Stokes equations in 3-D BFC systems. The main difference between these methods lies in the way of finding a pressure field such that the flowfield can be as close to divergence-free as possible (i.e. to satisfy the mass conservation equation). This is the main feature and difficulty of solving the incompressible flow problems. Numerical methods of References 1, 2 and 3, for instance, have employed the pseudocompressibility approach and time-iterative scheme to generate the pressure field so that the continuity equation is satisfied when a steady state solution is reached. In these methods, artificial smoothing techniques must be used to obtain a strong coupling between the velocity and pressure fields. Methods of References 4, 5 and 6, on the other hand, have utilized a successive pressure-velocity correction scheme by using a Poisson's equation for pressure correction derived approximately form the continuity and momentum equations. For these latter methods, grid staggering between the velocity vectors and the pressure nodes must be used to ensure stability of the numerical solutions.

There are several possible methods of grid staggering associated with different features in solving the pressure correction equation. These grid staggering methods were discussed in Reference 6, from which one of the methods was shown to be the most promising arrangement (i.e. with the velocity vectors located at the faces of a volume which contains the pressure and other scalars at its center). But, this method has one drawback, that the velocity components are solved using different control volumes. It is for this reason that a grid staggering system similar to the one used by Vanka et al. [4] is developed in the present study. The present method of grid staggering and pressure correction equation that was described by Vanka [4] and Maliska [6]. Also, using the present method, the same control volume is used for the velocity components and scalar quantities.
In the following sections, basic elements for establishing the present computer code for solving the curvilinear Navier-Stokes equations in three-dimensional space (CNS3D) will be described. These are followed by a series of standard numerical examples used to evaluate the accuracy and efficiency of the present numerical method. The numerical examples include laminar flow driven-cavity problem, cases of laminar and turbulent flows over backward-facing steps, and 3-D laminar flows inside a 90-deg-bend square duct. Applications of the present code to the internal flow problems of SSME will be included in future publications.

A user's guide to the present CNS3D code is provided in Appendix A. Appendix B contains a list and definitions of all the major fortran symbols used in the computer program which is listed in Appendix C.

TRANSFORMATION OF THE EQUATIONS OF MOTION

For incompressible Newtonian fluid, the continuity, momentum and energy equations can be written as:

\[
U_t + E_x + F_y + G_z = S
\]  

where \((x,y,z)\) represent the Cartesian coordinates, and

\[
U = \begin{bmatrix}
\rho \\
\rho u \\
\rho v \\
\rho w \\
\rho e - Q
\end{bmatrix} \quad \quad \begin{align*}
\rho u \\
\rho u u - \mu u_x \\
\rho u v - \mu v_x \\
\rho u w - \mu w_x \\
\rho u e - k'T_x
\end{align*}
\]

\[
E = \begin{bmatrix}
\rho v \\
\rho v u - \mu v_y \\
\rho v w - \mu w_y \\
\rho v e - k'T_y
\end{bmatrix} \quad \quad \begin{align*}
\rho w \\
\rho w u - \mu w_z \\
\rho w v - \mu w_z \\
\rho w e - k'T_z
\end{align*}
\]
\[
S = \begin{cases}
0 \\
(u u_x)_x + (u v_x)_y + (u w_x)_z - P_x \\
(u u_y)_x + (u v_y)_y + (u w_y)_z - P_y \\
(u u_z)_x + (u v_z)_y + (u w_z)_z - P_z \\
\mu [2(u_x^2 + v_y^2 + w_z^2) + (v_x + u_y)^2 + (w_y + v_z)^2 + (u_z + w_x)^2 - \frac{2}{3} (u_x + v_y + w_z)^2]
\end{cases}
\]

\[e = \text{the internal energy per unit mass} = C_v T \text{ for perfect gas}\]

\[Q = \text{energy added per unit volume}\]

\[k' = \text{thermal conductivity of the fluid}\]

Equation (1) is transformed to a general curvilinear coordinate system \((\xi, \eta, \zeta)\), which results in equation (2).

\[
U_t + E_\xi \xi_x + E_\eta \eta_x + E_\zeta \zeta_x + F_\xi \xi_y + F_\eta \eta_y + F_\zeta \zeta_y \\
+ G_\xi \xi_z + G_\eta \eta_z + G_\zeta \zeta_z = S
\]

where

\[
\xi_x = J(y_\eta z_\zeta - y_\zeta z_\eta) \\
\xi_y = -J(x_\eta z_\zeta - x_\zeta z_\eta) \\
\xi_z = J(x_\eta y_\zeta - x_\zeta y_\eta) \\
\eta_x = -J(y_\xi z_\zeta - y_\zeta z_\xi) \\
\eta_y = J(x_\xi z_\zeta - x_\zeta z_\xi) \\
\eta_z = -J(x_\xi y_\zeta - x_\zeta y_\xi)
\]
\[ \zeta_x = J(y_{\xi} z_\eta - y_\eta z_{\xi}) \]
\[ \zeta_y = -J(x_{\xi} z_\eta - x_\eta z_{\xi}) \]
\[ \zeta_z = J(x_{\xi} y_\eta - x_\eta y_{\xi}) \]
\[ J = \frac{1}{\left[ x_{\xi}(y_\eta z_{\xi} - y_{\xi} z_\eta) - x_\eta(y_{\xi} z_\eta - y_\eta z_{\xi}) + x_{\xi}(y_\eta z_{\xi} - y_{\xi} z_\eta) \right]} \]

The transformation coefficients, \( \xi_x, \xi_y, \xi_z, \eta_x, \eta_y, \eta_z, \zeta_x, \zeta_y, \) and \( \zeta_z, \) are computed numerically using second order central differencing. In the transformed domain, the grid sizes (i.e., \( \Delta \xi, \Delta \eta, \) and \( \Delta \zeta \)) are set to be unity. This simplifies the calculation of the transformation coefficients.

For turbulent flow computations, the present code has employed the standard \( k-\epsilon \) turbulence model [7] to provide the turbulent eddy viscosity \( \mu_t. \) The standard \( k-\epsilon \) turbulence model (which consists of a turbulent kinetic energy equation, \( k-\) equation, and a turbulent kinetic energy dissipation rate equation, \( \epsilon-\)equation) is given as:

\[ (\rho k)_t + \left( \rho u_1 k - \frac{\mu_{\text{eff}}}{\sigma_k} k x_i x_i \right) x_i = \rho (P_r - \epsilon) \] (3)

\[ (\rho \epsilon)_t + \left( \rho u_1 k - \frac{\mu_{\text{eff}}}{\sigma_\epsilon} \epsilon x_i x_i \right) x_i = \rho \frac{\epsilon}{k} (C_1 P_r - C_2 \epsilon) \] (4)

where the effective viscosity \( \mu_{\text{eff}} \) is calculated from:

\[ \mu_{\text{eff}} = \mu + \mu_t = \mu + \rho C_\mu \frac{k^2}{\epsilon} \]

and the turbulent kinetic energy production term, \( P_r, \) is defined as:

\[ P_r = C_\mu \frac{k^2}{\epsilon} \left[ (u_y + v_x)^2 + (v_z + w_y)^2 + (w_x + u_z)^2 + 2(u_x^2 + v_y^2 + w_z^2) \right] \]

The turbulence model constants are:

\[ C_\mu = 0.09, \quad C_k = 1.0, \quad C_\epsilon = 1.3 \]
\[ C_1 = 1.44, \quad C_2 = 1.92 \]
Also, the molecular viscosity $\mu$ in equation (1) is replaced by the effective viscosity $\mu_{\text{eff}}$ for turbulent flow cases.

In order to save the computational efforts, the widely used wall function approach [8] is employed to provide the near wall boundary conditions for the momentum and energy equations and the $k-\varepsilon$ turbulence model. This approach avoids the requirement of integrating the governing equations up to the wall which requires a large number of additional grid points near the wall.

Equations (2), (3), and (4) form a closed set of nonlinear partial differential equations governing the fluid motion. This set of equations are to be solved by means of finite difference approximations which are performed in the transformed domain. For treating the convection terms, the hybrid scheme [9] is employed for simplicity (although other more elaborate schemes such as central differencing plus artificial dissipation scheme, QUICK scheme, or skew upwind differencing scheme, etc. can be implemented [10]). These are described in the following sections.

**DISCRETIZATION OF THE EQUATIONS OF MOTION**

In this section, finite difference approximations are used to discretize the governing equations, equations (2), (3), and (4). Second-order central differencing is used for the diffusion terms and the source terms. The hybrid differencing scheme [9] is employed to approximate the convection terms in the governing equations. The finite difference discretizations are performed in the transformed domain. The solution procedure for the discretized equations using a velocity-pressure correction algorithm (SIMPLE-C) of References 11 and 12 will be described in the next section.

The governing equations of motion can be represented by the following model transport equation in which $\phi$ denotes all the dependent variables respectively and $\Gamma$ is the diffusion coefficient.

\[
\begin{align*}
(r \phi)_t + & [\rho u \phi - \Gamma (\phi \xi_x + \phi \eta_x \xi_x + \phi \varepsilon_x \xi_x)] \xi_x \\
& + [\rho u \phi - \Gamma (\phi \xi_x + \phi \eta_x \xi_x + \phi \varepsilon_x \xi_x)] \eta_x \\
& + [\rho u \phi - \Gamma (\phi \xi_x + \phi \eta_x \xi_x + \phi \varepsilon_x \xi_x)] \xi_x \\
& + [\rho v \phi - \Gamma (\phi \xi_y + \phi \eta_y \xi_y + \phi \varepsilon_y \xi_y)] \xi_y \\
& + [\rho v \phi - \Gamma (\phi \xi_y + \phi \eta_y \xi_y + \phi \varepsilon_y \xi_y)] \eta_y \\
& + [\rho v \phi - \Gamma (\phi \xi_y + \phi \eta_y \xi_y + \phi \varepsilon_y \xi_y)] \xi_y \\
& + [\rho w \phi - \Gamma (\phi \xi_z + \phi \eta_z \xi_z + \phi \varepsilon_z \xi_z)] \xi_z \\
& + [\rho w \phi - \Gamma (\phi \xi_z + \phi \eta_z \xi_z + \phi \varepsilon_z \xi_z)] \eta_z \\
& + [\rho w \phi - \Gamma (\phi \xi_z + \phi \eta_z \xi_z + \phi \varepsilon_z \xi_z)] \xi_z = S .
\end{align*}
\]  \hspace{1cm} (5)
Discretization of equation (5) is performed using finite difference approximations in the transformed domain. The second order central differencing is used for approximating the diffusion terms. For the convection terms, the hybrid differencing scheme [9] is employed (i.e., using central differencing for cell Peclet number less than or equal to 2 and switching to upwind differencing when the cell Peclet number is greater than 2). The finite difference equation is arranged by collecting terms according to the grid nodes around a control volume as shown in Figure 1. The final expression is given by equation (6) in which $A$ represents the link coefficients between grid nodes P, E, W, N, S, T, B, NE, NW, NT, NB, SE, SW, ST, SB, ET, EB, WT, and WB as shown in Figure 1.

$$A_P \phi_P = A_E \phi_E + A_W \phi_W + A_N \phi_N + A_S \phi_S + A_T \phi_T + A_B \phi_B + S_1$$ (6)

where

$$S_1 = S + A_P^o \phi_P^o + A_{NE} \phi_{NE} + A_{NW} \phi_{NW} + A_{NT} \phi_{NT} + A_{NB} \phi_{NB}$$
$$+ A_{SE} \phi_{SE} + A_{SW} \phi_{SW} + A_{ST} \phi_{ST} + A_{SB} \phi_{SB}$$
$$+ A_{ET} \phi_{ET} + A_{EB} \phi_{EB} + A_{WT} \phi_{WT} + A_{WB} \phi_{WB}$$

$$A_P = A_E + A_W + A_N + A_S + A_T + A_B + A_P^o$$

$$A_P^o = \rho_P^o/\Delta t$$

The subscript $^o$ denotes the solution at the previous time level. A fully implicit formulation is employed for solving the time dependent transient problems.

Figure 1. Three-dimensional grid structure and labeling around a grid node P.
Thus, the nonlinear equations of motion are approximated by a system of linear algebraic equations which have the form of equation (6). Only one program subroutine is designed to calculate the link coefficients and the source terms. The number of algebraic equations depends on the number of interior grid points. For a grid size of 10 x 10 x 10 the number of algebraic equations to be solved would be around 512. This large system of equations are preferred to be solved by some iterative methods, such as Gauss-Seidel iteration, line-underrelaxation method [13] or Stone's method [14], etc., rather than using direct methods such as Gaussian elimination method. Only a few (6 to 10) iterations through the whole computational domain are needed and a complete convergence of the system of algebraic equations is not required. Since equation (6) is only a linearized version of the governing equations which are nonlinear and coupled in nature, solutions of the equations of motion must be obtained through global iterations among the equations. A tentative solution to equation (6) will not affect the final results significantly. On the other hand, if too many iterations are used to get a better solution of equation (6), then a great deal of computing time would be virtually wasted. However, the above argument cannot be applied when the pressure correction equation (which will be derived in the next section) is solved. Since during each global iteration it is desirable to retain a divergence-free velocity field, better solution of the pressure correction equation would in effect promote the convergence of the whole numerical scheme. Therefore, more iterations are usually used to solve the pressure correction equation.

SOLUTION PROCEDURES

The governing equations used in the present analysis are nonlinear and strongly coupled. Iterative procedures are employed to drive the equations to a converged solution. It is particularly important for incompressible flow to make the flow field satisfy the continuity equation and the momentum equations at the same time. This requires a correct pressure field associated with a divergence-free velocity field. A velocity-pressure correction procedure is developed in the present study to drive the pressure field and the velocity field to be divergence free. This kind of procedure requires grid staggering between the velocity components and the locations where the pressure is estimated and stored such that the velocity field and the pressure field will not be uncoupled.

In the present study, staggering grid systems as shown in Figure 2 (for 2-D case) are used. The velocity components, u and v, are solved and stored at the grid nodes and the pressure, p, is located at the corners of the control volume of u and v. In this way, solutions of u and v can be solved using the same control volume and coupling between u, v and p can also be enforced. To estimate the pressure field, a pressure correction equation is derived approximately from the discretized momentum and continuity equations. The velocity and pressure fields are then corrected using the solutions of the pressure correction equation.

First, the finite difference momentum equations (for u, v and w) can be written as:

\[ A_p u_p^* = \sum A_i^u u_i^* = p_x^* + S_u \]  

\[ (8a) \]
Figure 2. Locations where the variables are stored (staggering grids are used).

\[ A_p^V \sum A_i^V v_i^* = P_y^* + S_v \]  \hspace{1cm} (8b)

\[ A_p^W \sum A_i^W w_i^* - P_z^* + S_w \]  \hspace{1cm} (8c)

where \( u^*, v^*, w^*, \) and \( p^* \) represent the solutions of equations (8a) and (8b). To satisfy the continuity equation the velocities and pressure are corrected according to the following relations:

\[ u = u^* + u' \]  \hspace{1cm} (9a)

\[ v = v^* + v' \]  \hspace{1cm} (9b)

\[ w = w^* + w' \]  \hspace{1cm} (9c)

\[ p = p^* + p' \]  \hspace{1cm} (9d)
A new set of momentum equations can be constructed approximately using the divergence-free flow field, \( u, v, w, \) and \( p \):

\[
A_p^u u_p = \sum_i A_i^u u_i - P_x + S_u \tag{10a}
\]

\[
A_p^v v_p = \sum_i A_i^v v_i - P_y + S_v \tag{10b}
\]

\[
A_p^w w_p = \sum_i A_i^w w_i - P_z + S_w \tag{10c}
\]

By subtracting equations (8a) through (8c) from equations (10a) through (10c), respectively, the following equations result:

\[
A_p^u u_p' = \sum_i A_i^u (u_i' - u_p') - P_x' \tag{11a}
\]

\[
A_p^v v_p' = \sum_i A_i^v (v_i' - v_p') - P_y' \tag{11b}
\]

\[
A_p^w w_p' = \sum_i A_i^w (w_i' - w_p') - P_z' \tag{11c}
\]

According to SIMPLE-C algorithm [11], equations (11a) through (11c) are rearranged to be:

\[
(A_p^u - \sum_i A_i^u) u_p' = \sum_i A_i^u (u_i' - u_p') - P_x' \tag{12a}
\]

\[
(A_p^v - \sum_i A_i^v) v_p' = \sum_i A_i^v (v_i' - v_p') - P_y' \tag{12b}
\]

\[
(A_p^w - \sum_i A_i^w) w_p' = \sum_i A_i^w (w_i' - w_p') - P_z' \tag{12c}
\]
The first terms on the right-hand side of equations (12a) through (12c) are neglected to simplify the formulation. Thus,

\[
\begin{align*}
    u'_p &= - \left( \frac{1}{A_p u} - \sum_i A_i u \right) P'_x = - D_u P'_x \quad (13a) \\
    v'_p &= - \left( \frac{1}{A_p v} - \sum_i A_i v \right) P'_y = - D_v P'_y \quad (13b) \\
    w'_p &= - \left( \frac{1}{A_p w} - \sum_i A_i w \right) P'_z = - D_w P'_z \quad (13c)
\end{align*}
\]

Using the decompositions of equations (9a) through (9c), the continuity equation can be written as:

\[
    u_x + v_y + w_z = (u_x^* + v_y^* + w_z^*) + (u'_x + v'_y + w'_z) = 0 \quad . (14)
\]

Substituting equations (13a) through (13c) into equation (14), the following pressure correction equation can be obtained:

\[
    \left[ (D_u P'_x)'_x + (D_v P'_y)'_y + (D_w P'_z)'_z \right] = -(u_x^* + v_y^* + w_z^*) \quad . (15)
\]

Equation (15) is a Poisson's equation with the source term equal to the local divergence of the flow field. To enforce the coupling between the velocity and pressure fields, the source term of equation (15) is first evaluated at the control volumes centered between the velocity nodes as shown in Figure 3. An averaged source term is then calculated at the cell center of p node for solving equation (15). In this way, the difficulties in solving the pressure correction equation, as described by Vanka [4] and Maliska [6], are eliminated. Coupling between the velocity and the pressure field is also assured.

According to the above analyses, the present numerical method contains the following solution steps:

1) Guess initial velocity and pressure field.
2) Solve for the velocity field using equations (8a) through (8c).
3) Solve for other scalar transport equations.
4) Solve the pressure correction equation, equation (15).
Figure 3. Control volumes where the mass conservation is evaluated for solving the pressure correction equation.

5) Correct the velocity and pressure fields using equations (13a) through (13c) and equation (9d).

6) Go back to step (2) until solution converges.

A converged solution is obtained when the following criterion is met:

\[
\text{Error} = \left( |\Delta u|_{\text{max}} + |\Delta v|_{\text{max}} + |\Delta w|_{\text{max}} \right) / U_{\text{ref}} + |P'|_{\text{max}} / \rho U_{\text{ref}}^2 \\
\leq 3 \times 10^{-4}
\]

where \(\Delta u\), \(\Delta v\), and \(\Delta w\) represent velocity changes during each iteration due to the solutions of the momentum equations.

In solving the momentum equations in step (b) above, underrelaxation factor of about 0.6 is recommended. With this, \(A_p\)'s in equations (8a) through (8c) are modified according to the underrelaxation factor. For the correction of velocity field, no underrelaxation is required. But the correction of pressure field should be underrelaxed slightly (around 0.9) when the grid nonorthogonality is strong. This is different from that suggested by References 11 and 12 (which recommend no underrelaxation for pressure correction).
NUMERICAL EXAMPLES

In this section, several numerical examples are employed to demonstrate the efficiency and accuracy of the present numerical method. To serve this purpose, 2-D and 3-D, laminar and turbulent flow cases are included. These cases are: (a) 2-D laminar driven square-cavity flows; (b) 2-D laminar flows over a backward-facing step; (c) 2-D turbulent flows over a backward-facing step; (d) 3-D developing laminar flow inside a 90-deg-bend square duct. Detailed descriptions and results of the computation of the above cases are included as follows.

A. 2-D Laminar Driven Square-Cavity Flows

The first test case is concerning laminar recirculating flows inside a square cavity. Only one side of the walls is moving at a constant speed tangent to that wall. This case has been studied extensively by Burggraf [15] and has often been used as one of the standard testing cases for numerical methods in solving the incompressible Navier-Stokes equations. Physical geometry and wall boundary conditions are illustrated in Figure 4. Reynolds number of the flow (based on the cavity size and the moving wall velocity) studied in the present analysis is 400. Two different mesh systems, as shown in Figure 5, are used to study the effect of grid non-orthogonality on the accuracy of the present method. The grid system of Figure 5(a) is uniform and orthogonal while the grid system of Figure 5(b) is non-uniform and non-orthogonal.

![Figure 4. Physical geometry and wall boundary conditions for laminar flows inside a wall-driven square cavity.](image-url)
Figure 5. Mesh systems used for driven cavity problem. (a) Uniform and orthogonal grid. (b) Nonuniform and nonorthogonal grid.
Results of the computations are shown in Figures 6 and 7. Velocity vector plots of the predicted flow fields are compared in Figure 6 for the mesh systems shown in Figure 5. Detailed comparisons of the predicted velocity profiles along the mid-section of the cavity are illustrated in Figure 7. Predicted results of Burggraf [15] are also included. Good agreements between the present calculations and those
of Burggraf [15] are also included. Good agreements between the present calculations and those of Burggraf are shown in Figure 7. Discrepancies between the present predictions and Burggraf's results are mainly due to the hybrid differencing scheme used in the present method. The upwind part of the hybrid scheme produces large numerical diffusion which tends to reduce the strength of the vorticity inside the cavity. Effects of differencing schemes in approximating the convection terms on the predicted results will be studied in the next test case.

Convergence history of the computation of the present case using uniform grids is given in Figure 8 which shows that the present numerical method is quite different. Almost identical convergence rates were found for the non-orthogonal case.

![Figure 8](image.png)

Figure 8. Convergence history for the driven cavity problem, Re = 400.

B. 2-D Laminar Flows Over a Backward-facing Step

This test case concerns 2-D laminar recirculating flows over a backward-facing step with 1:2 expansion ratio. The dependence of the size of the recirculation region (characterized by the reattachment length) on the Reynolds number (based on the inlet bulk velocity and twice of the inlet channel width) of the flow is of major concern. The physical domain and boundary conditions are illustrated in Figure 9 in which a fully developed laminar flow velocity profile is imposed at the flow entrance. A non-uniform grid of 45 x 45 was used for numerical computations. Several cases with different Reynolds numbers from 100 to 800 have been studied. An experimental and theoretical study about this problem, which results will be used as the basis of data comparisons, has been provided by Amaly et al. [16].
To save computational efforts, the solution of one case with Reynolds number 100 is obtained in the first run. Then, a series of cases with increasing Reynolds numbers (i.e., 100, 200, 300, 400, 600, and 800) are calculated using the preceding results of lower Reynolds number as the initial guesses of the flow field. In this way, an average of 500 iterations for each case were needed to obtain converged solutions.

Two different differencing schemes in approximating the convection terms are employed to demonstrate the effects of the differing schemes on the predictions. One of the schemes is the widely used hybrid scheme [9]. The other scheme employs the central differencing scheme plus an artificial dissipation term used to stabilize the solution which is similar to the one used by Rhie [17]. The artificial dissipation term becomes effective only when the cell Peclet number (or cell Reynolds number) exceeds 10.

Results of the present predictions using two different differencing schemes are compared with the experimental measurements [16] and other predictions as shown in Figure 10. It can be seen clearly from Figure 10 that the present method with hybrid scheme gives results similar to those predicted by TEACH code [16] while the present method with central differencing and artificial dissipation reveals predictions close to those predicted by INS3D [18] and the method of Kim and Moin [19]. This is reasonable since the TEACH code and the present method (with the first scheme) use the hybrid scheme which introduces large numerical dissipation by its upwind part (for cell Peclet number greater than 2). This tends to reduce the reattachment length for Reynolds number greater than 400. The second scheme, which is similar to the ones used in INS3D and the method of Kim and Moin, has the numerical accuracy close to second order by setting the artificial dissipation to be as small as the solution stability permits such that better accuracy of the predictions is expected.
Figure 10. Reattachment length versus Reynolds number for laminar flows over a backward-facing step (1:2 expansion).

Stream function plots of the predictions using the two differencing schemes for Reynolds number 600 are compared in Figure 11. It is shown in Figure 11 that the second scheme gives a smooth shape of the recirculation zone while the hybrid scheme gives a sudden change in the shape of the recirculation region upstream of the reattachment point. Also, larger sizes of the separation regions on the step side wall and along the upper wall are predicted using the second scheme.

Figure 11. Streamline plots for laminar flow over a backward-facing step (1:2 expansion). (a) Hybrid Scheme. (b) Central differencing plus artificial dissipation scheme.
C. 2-D Turbulent Flows Over a Backward-Facing Step

In order to demonstrate the applicability of the present method to turbulent flow case, one of the standard test cases presented in the Stanford Conference [20] is selected here (i.e., turbulent flow over a 2:3 expansion backward-facing step). The standard k-ε turbulence model was used to provide the eddy viscosity for the transport equations. The physical geometry and boundary conditions imposed are shown in Figure 12. The calculation domain extends upstream of the expansion plane by 4 step heights and downstream of the expansion plane by 30 step heights to assure a fully developed velocity profile at the exit. A uniform velocity profile is located at the inlet plane. A 45 x 42 grid was used in the computation. 300 iterations were required to obtain converged solutions. Only hybrid differencing schemes were used in this case.

![Figure 12. Physical geometry and boundary conditions of turbulent flows over a backward-facing step (2:3 expansion).](image)

Results of the computation are shown in Figures 13, 14, and 15. These results are compared with the experimental measurements [20]. The under-prediction of the reattachment length is mainly due to the fast development of the mixing layer downstream of the expansion plane which is the characteristics of the standard k-ε turbulence model. Numerical diffusion provided by the hybrid scheme also contributes some part to the discrepancies between the predictions and measurements.

D. Developing Laminar Flow Inside a 90-Deg-Bend Square Duct

This test case simulates a three-dimensional developing laminar flow inside a 90-deg-bend square duct as illustrated in Figure 16(a). The symmetry plane is located at z = 0 where the symmetric boundary conditions are imposed. A fully developed velocity profile of laminar flow inside a straight square duct is prescribed at the entrance which is 2.8 duct widths upstream of the bend. A zero pressure
Figure 13. Locus of flow reversal inside the recirculation region for turbulent flow over a backward-facing step (2:3 expansion).

Figure 14. Stream line pattern of turbulent flow over a backward-facing step with 2:3 expansion ratio.

Figure 15. Contours of turbulent kinetic energy ($k/U_o^2$) of turbulent flow over a backward-facing step with 2:3 expansion ratio.
Figure 16. Geometry and mesh system of a 90-deg-bend square duct developing laminar flow problem.

Gradient exit (which is 4.5 duct widths downstream of the bend) boundary condition is imposed. The Reynolds number of the flow (based on the duct hydraulic diameter and the inlet bulk velocity) is 790. A 21 x 18 x 10 grid was used for numerical computations. The front view and side view of the mesh system are illustrated in Figure 16(b). Experimental measurements of Humphrey et al. [21] are used for data comparisons.

Velocity vector plots on three sections along the main flow directions (i.e., on x-y plane) are shown in Figure 17. Secondary flow patterns at several stations across the bend are illustrated in Figure 18. These results are very similar to those obtained by Vanka [22] and Rhie [23]. Grid sizes of 50 x 22 x 15 and 58 x 15 x 11 were used by Rhie and Vanka, respectively. The present investigation, using only less than half of their grid numbers, gives highly encouraging results. Detailed comparisons between the measured and the predicted main velocity profiles are given in Figure 19.

With the above successful numerical simulations, it is believed that the present numerical method can be applied to general fluid dynamics problems with good numerical accuracy and efficiency.
Figure 17. Primary velocity patterns of laminar flow inside a 90-deg-bend square duct.

(a) z/d = 0.0, (b) z/d = 0.25, (c) z/d = 0.48.
Figure 18: Secondary velocity patterns of laminar flow inside a 90-deg bend.
(a) $\theta = 0$ deg. (b) $\theta = 30$ deg. (c) $\theta = 60$ deg. (d) $\theta = 90$ deg.
Figure 19. Primary velocity profiles for a 3-D 90-deg-bend square duct. (a) z/d = 0.0. (b) z/d = 0.25.

CONCLUSIONS

A numerical method for solving the steady or transient incompressible Navier-Stokes equations in three-dimensional body-fitted coordinate systems has been developed. In the present paper, the basic numerical algorithms and grid arrangements have been described in detail. A brief user's guide to the present computer code (CNS3D) has been included in Appendix A. A program listing has also been attached in Appendix C.

Several numerical testing examples of 2-D and 3-D, laminar and turbulent flow problems included in the present work have demonstrated that the present computer code is efficient and robust, and can be used as a reliable tool for engineering design and analysis applications. Applications of the present code to the internal turbulent flow problems of the SSME will be presented in the future publications.
REFERENCES


24


The global structure of the present computer code (CNS3D) can be represented by a flow chart, shown in Figure A-1. The user is referred to Appendix C for detailed information. First, the program requests inputs, from logic unit 5 (LU = 5), of program control parameters that specify the maximum number of iterations, the type of flow (i.e., laminar or turbulent), number of iterations for solving the pressure correction equation (typically 10), and underrelaxation factors for solving the transport equations, etc. This is followed by the definitions of all the program constants including turbulence model constants (these constants are subject to change according to the user's specific flow problem). Next, the program asks for inputs of the initial flow field guess from a restart file (LU = 8) which contains the grid system coordinates and flow field data that may be created by the user (including grid generation) or obtained from the previous solutions. Format of this data file is also subject to change according to the user's preference. Next, wall boundary control parameters, boundary grid normal distance to the wall, and wall boundary direction cosine are calculated in subroutine DIRCOS. Subroutine TRANF is then invoked to obtain the grid transformation coefficients. Before the solution procedure starts, the inlet mass flow rate is calculated which will be used to control the outlet mass flow rate to enhance mass conservation. The solution procedures consist of a series of subroutine calls to SOLVEQ starting from the solutions of the velocity vectors, u, v, and w, and then the solutions of scalar quantities (including the energy equation and the turbulence model equations) and finally the solution of the pressure correction equation to update the velocity and pressure field such that a divergence-free flow field can be retained.

After each global iteration of the solution procedures, the numerical of iterations and the maximum flow field corrections are checked with the initial settings. If the convergence criterion is satisfied or the number of iterations reaches the prescribed value then the solution procedures stop and the flow field solutions will be written on the pre-assigned disc file (LU = 7).

For instance, if a steady-state laminar flow problem (Reynolds number of 600) is of interest and a converged solution is expected within 300 iterations and the number of iterations for solving the pressure correction equation is 10 and the underrelaxation factors are 0.5 and 0.95 for transport equations and pressure correction equation, respectively, the first inputs from LU = 5 would be:

<table>
<thead>
<tr>
<th>Line</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>300</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.95</td>
</tr>
<tr>
<td>3.</td>
<td>600</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the second input sequence (i.e., from restart file), the program reads in L x M x N lines of data records. See Figure A-2 for grid structures. Notice that the program requires variable dimensions of (L+1, M+1, N+1) for solving the pressure correction equation. It is important to check the COMMON table for proper variable dimensions.
Figure A-1. Global structure of the present computer program CNS3D.
If the flow problem involves symmetric or cyclic boundary conditions, then the user can look into the subroutine SYMOUT to specify the appropriate boundary conditions (the conditions shown in the program listing of Appendix C are for symmetric boundary conditions at \( K = 1 \)). For cyclic boundary conditions at \( K = 1 \) and \( K = N \), data at \( K = 2 \) and \( K = N-1 \) can be used to obtain boundary conditions at \( K = 1 \) and \( K = N \) by requiring same gradients across \( K = 1 \) and \( K = N \). This method is simple but will lag the boundary conditions by one iteration. A direct method without lagging the boundary conditions can also be employed by modifying the subroutine of linear algebra solver LINERX such that the boundary conditions can be part of the solution of the TDMA (tridiagonal matrix) solver.

In case of incorporating different wall functions for turbulent flow problems (e.g., References 24, 25, and 26), subroutine BOUNC and WALFN can be modified according to the user's method of wall treatments. The set of wall functions given in the program listing of Appendix C are derived from the conventional wall law and the equilibrium turbulent kinetic energy relations [8].

When additional source terms are to be added to the transport equations due to flow problem requirements, modifications to the source term calculation section in the subroutine SOLVEQ can be carried out. Notice that in the subroutine SOLVEQ source terms for the velocities \( v \) and \( w \) are included in the \( u \)-source section. Purpose of this is to save some computing time since these source terms use similar calculation routines.

Some times it is required to solve more transport equations other than the basic ones included in Appendix C. To modify the program to incorporate more equations,
several changes are necessary. First, new variables must be added to the COMMON table (this can be easily done through the computer editor session). Then, new source term sections are added in the SOLVEQ subroutine. Finally, subroutines WALFN and SYMOUT are modified to incorporate the new variables into the boundary condition setting routines.
APPENDIX B

LIST OF FORTRAN SYMBOLS

A(K) = Matrix elements of a tridiagonal matrix
AB(I,J,K) = Link coefficients through the bottom face of a control volume
AE(I,J,K) = Link coefficients through the east face of a control volume
ALC = Underrelaxation factor for symmetry or cyclic boundary conditions
ALE = Underrelaxation factor for the $\varepsilon$-equation
ALK = Underrelaxation factor for the $k$-equation
ALP = Underrelaxation factor for the pressure correction equation
ALU = Underrelaxation factor for the $u$-equation
ALV = Underrelaxation factor for the $v$-equation
ALVIS = Underrelaxation factor for the effective viscosity
ALW = Underrelaxation factor for the $w$-equation
AN(I,J,K) = Link coefficient through the north face of a control volume
ANAB = Sum of the link coefficients at all faces
ANV1(I) = Modified wall boundary link coefficient for $v$-equation
ANW1(I) = Modified wall boundary link coefficient for $w$-equation
AP(I,J,K) = Sum of the link coefficients around a control volume
APO(I,J,K) = Link coefficients in time marching direction
ARDEN = Area times density across a section in physical domain
AREA = Area of a section in physical domain
AS(I,J,K) = Link coefficients through the south face of a control volume
AT(I,J,K) = Link coefficients through the top face of a control volume
AW(I,J,K) = Link coefficients through the west face of a control volume
B(K) = Matrix elements of a tridiagonal matrix
BB(I,J,K) = Coefficients in Stone's partial factorization technique
BOUNC = Subroutine for getting turbulent wall boundary conditions through wall functions
C(K) = Matrix elements of a tridiagonal matrix

C1 = Turbulence model constant, $= 1.44$

C2 = Turbulence model constant, $= 1.92$

CB = Convective flux through the bottom face of a control volume

CE = Convective flux through the east face of a control volume

CK = Von Karman constant, $= 0.4$

CMU = Turbulence model constant, $= 0.09$

CMU1 = CMU**0.25

CMU2 = CMU**0.75

CN = Convective flux through the north face of a control volume

CS = Convective flux through the south face of a control volume

CT = Convective flux through the top face of a control volume

CW = Convective flux through the west face of a control volume

CX(I,J,K) = Grid transformation coefficient, $\xi_x$

CY(I,J,K) = Grid transformation coefficient, $\xi_y$

CZ(I,J,K) = Grid transformation coefficient, $\xi_z$

D(K) = Matrix elements of a tridiagonal matrix

DDB = Diffusive flux through the bottom face of a control volume

DDE = Diffusive flux through the east face of a control volume

DDN = Diffusive flux through the north face of a control volume

DDS = Diffusive flux through the south face of a control volume

DDT = Diffusive flux through the top face of a control volume

DDW = Diffusive flux through the west face of a control volume

DE(I,J,K) = Turbulent kinetic energy dissipation rate, $\varepsilon$

DEO(I,J,K) = DE at the previous time level

DEN(I,J,K) = Density of the fluid

DENO(I,J,K) = DEN at the previous time level

DENC = Density at the center of a surface
DENIN = Initial value of density of the fluid
DIRCOS = Subroutine for calculating the boundary grid sizes and direction cosines
DITM = Wall boundary average value of dissipation rate
DK(I,J,K) = Turbulent kinetic energy, k
DKO(I,J,K) = DK at the previous time level
DTT = Time step size, Δt
DU(I,J,K) = Diffusive coefficient for the p'-equation
DV(I,J,K) = Diffusive coefficient for the p'-equation
DW(I,J,K) = Diffusive coefficient for the p'-equation
E = Wall law constant, = 9.01069
EREEXT = Convergence criterion tolerance
ERRE = Maximum correction in ε
ERRF = Maximum correction of a variable
ERRK = Maximum correction in k
ERRM = Maximum correction in p
ERRU = Maximum correction in u
ERRV = Maximum correction in v
ERRW = Maximum correction in w
EX(I,J,K) = Grid transformation coefficient, \( \eta_x \)
EY(I,J,K) = Grid transformation coefficient, \( \eta_y \)
EZ(I,J,K) = Grid transformation coefficient, \( \eta_z \)
F(I,J,K) = Tentative variable of the transport equations
FO(I,J,K) = F at the previous time level
F1(I,J,K) = Variable quantity at the previous iteration step
FLOW = Outlet mass flow rate
FLOWIN = Inlet mass flow rate
GEN(I,J,K) = Turbulent kinetic energy production rate
HNUM = Large number, = 1.E30
I = Index along the ξ grid lines
IBC(I) = Boundary grid index
IE = Index assigned for the transport equations
IG = Problem control parameter, =1 for laminar flow and =2 for turbulent flow
IITO = Total number of wall boundary grids
ITY = Boundary grid face type
IJLO(I,J,K) = Boundary grid sequential order
INIT = Subroutine for initializing variables
INPRO = Logical parameter for updating the effective viscosity
INSOE = Logical parameter for solving the ε-equation
INSOK = Logical parameter for solving the k-equation
INSOP = Logical parameter for solving the p'-equation
INSOT = Logical parameter for solving the T-equation
INSOU = Logical parameter for solving the u-equation
INSOV = Logical parameter for solving the v-equation
INSOW = Logical parameter for solving the w-equation
IS = Starting value of I of the solution domain
ISWE = Number of sweeps for solving the ε-equation
ISWK = Number of sweeps for solving the k-equation
ISWP = Number of sweeps for solving the p'-equation
ISWU = Number of sweeps for solving the u-equation
ISWV = Number of sweeps for solving the v-equation
ISWW = Number of sweeps for solving the w-equation
IT = Last value of I of the solution domain
ITT = Number of time steps
J = Index along the η grid lines
JBC(I) = Boundary grid index
JS = Starting value of J of the solution domain
JT = Last value of J of the solution domain
K = Index along the \( \zeta \) grid lines
KBC(I) = Boundary grid index
KS = Starting value of K of the solution domain
KT = Last value of K of the solution domain
L = Maximum dimension of grid system in I direction
LO = L + 1
L1 = Starting point of blockage region in I direction
L2 = Last point of blockage region in I direction
LINERX = Subroutine for solving algebraic equations
LT = L - 1
M = Maximum dimension of grid system in J direction
MO = M + 1
M1 = Starting point of blockage region in J direction
M2 = Last point of blockage region in J direction
MC(I,J,K) = Wall blockage region control parameter
MT = M - 1
N = Maximum dimension of grid system in K direction
NO = N + 1
N1 = Starting point of blockage region in K direction
N2 = Last point of blockage region in K direction
NEWVIS = Subroutine for updating the effective viscosity
NLIMT = Limit of maximum number of iterations
NT = N - 1
P = Static pressure (relative)
PCXI = Pressure gradient, \( P_\xi \)
PDUV = Blockage control parameter for link coefficients
PEDA = Pressure gradient, $p_{n}$
PP = Pressure correction, $p'$
PPBLK = Global pressure correction
PSCI = Pressure gradient, $P_{\zeta}$
PTA = Wall boundary source term for the momentum equations
PW = Wall value control parameter
RENL = Reynolds number of the fluid
Sige = Turbulence model constant, $\varepsilon = 1.3$
SIGK = Turbulence model constant, $\kappa = 1.0$
SINX(I) = Wall boundary direction cosine
SINY(I) = Wall boundary direction cosine
SINZ(I) = Wall boundary direction cosine
SMNUM = Small number, $1.0 \times 10^{-30}$
SOC1 = Source term due to shear stress
SOC2 = Source term due to shear stress
SOC3 = Source term due to shear stress
SOLVEQ = Subroutine for solving general transport equation
SP(I,J,K) = Linear part of the source term
SPK(I,J,K) = Secondary linear part of the source term
SU(I,J,K) = Constant part of the source term
SUK(I,J,K) = Secondary constant part of the source term
SX(I,J,K) = Grid transformation coefficient, $\zeta_{x}$
SY(I,J,K) = Grid transformation coefficient, $\zeta_{y}$
SYMOUT = Subroutine for setting flow boundary conditions
SZ(I,J,K) = Grid transformation coefficient, $\zeta_{z}$
TAUN(I) = Wall shear stress
TIMT = Total time
TJO(I,J,K) = Jacobian of metric transformation
TM(I,J,K) = Temperature
TMO(I,J,K) = TM at the previous time level
TMULT = Wall shear stress
TRANF = Subroutine for calculating the grid transformation coefficients

TXXE(I,J,K) = Metric coefficient for east face diffusive flux
TXXW(I,J,K) = Metric coefficient for west face diffusive flux
TXYN(I,J,K) = Metric coefficient for north face diffusive flux
TXYS(I,J,K) = Metric coefficient for south face diffusive flux
TXZT(I,J,K) = Metric coefficient for top face diffusive flux
TXZB(I,J,K) = Metric coefficient for bottom face diffusive flux
TYYN(I,J,K) = Metric coefficient for north face diffusive flux
TYYS(I,J,K) = Metric coefficient for south face diffusive flux
TYXE(I,J,K) = Metric coefficient for east face diffusive flux
TYXW(I,J,K) = Metric coefficient for west face diffusive flux
TYZT(I,J,K) = Metric coefficient for top face diffusive flux
TYZB(I,J,K) = Metric coefficient for bottom face diffusive flux

TZXE(I,J,K) = Metric coefficient for east face diffusive flux
TZXW(I,J,K) = Metric coefficient for west face diffusive flux
TZYN(I,J,K) = Metric coefficient for north face diffusive flux
TZYS(I,J,K) = Metric coefficient for south face diffusive flux

U(I,J,K) = U-velocity
UO(I,J,K) = U at the previous time level
UC = Velocity at the center of a surface
UCXI = U-velocity gradient, $u_\xi$
UEDA = U-velocity gradient, \( u_\eta \)
UINC = Velocity correction at outlet plane
USCI = U-velocity gradient, \( u_\zeta \)
UX = U-velocity gradient, \( u_x \)
UY = U-velocity gradient, \( u_y \)
UZ = U-velocity gradient, \( u_z \)
V(I,J,K) = V-velocity
VO(I,J,K) = V at the previous time level
VISC = Molecular viscosity, \( \mu \)
VISE(I,J,K) = Effective viscosity, \( \mu_{eff} \)
VCXI = V-velocity gradient, \( v_\zeta \)
VEDA = V-velocity gradient, \( v_\eta \)
VSCI = V-velocity gradient, \( v_\zeta \)
VX = V-velocity gradient, \( v_x \)
VY = V-velocity gradient, \( v_y \)
VZ = V-velocity gradient, \( v_z \)
W(I,J,K) = W-velocity
WO(I,J,K) = W at the previous time level
WALLFN = Subroutine for calculating the wall functions
WALVAL = Subroutine for assigning wall values
WCXI = W-velocity gradient, \( w_\zeta \)
WEDA = W-velocity gradient, \( w_\eta \)
WSCI = W-velocity gradient, \( w_\zeta \)
WX = W-velocity gradient, \( w_x \)
WY = W-velocity gradient, \( w_y \)
WZ = W-velocity gradient, \( w_z \)
X(I,J,K) = X-coordinate
\[ Y(I, J, K) \]  = Y-coordinate
\[ YN(I) \]  = Wall normal distance from the last grid
\[ YN1(I) \]  = Wall grid volume size
\[ YPLN(I) \]  = Nondimensionalized \( YN \), \( y^+ = u_T y/\nu \)
\[ Z(I, J, K) \]  = Z-coordinate
FORTRAN VIII: LICENSED RESTRICTED RIGHTS AS STATED IN LICENSE ************, SEE DOCUMENTATION PACKAGE, QA-101M90.

1 000001I  PROGRAM CN530
2 C      SY: Y. S. CHEN  ON: 8/13/1985
3 C
4 C****( CLAVILINAR N-S CODE FOR 3-D INCOMPRESSIBLE FLOWS )**********
5 C
6 000006I  COMMON
7 1/VAR/(21,18,10),V(21,18,10),P(21,18,10),Q(21,18,10),
8 2/DE(21,18,10),ERR0,ERR0,ERRX,ERRX,ERRX,
9 3/PF(21,18,10),AW(21,18,10)TM(21,18,10),
10 1/PHI/.vec(21,18,10),CEN(21,18,10),VISC,DENIN,FLOWN
11 1/PCPA/ vec(21,18,10),OY(21,18,10),OX(21,18,10),
12 1/TURS/  SGE=SIGE,CMU1,CMU2,CMU3,CMU4,CMU5,CMU6,CMU7,
13 2/YN(800),YN(800),SINX(800),SINX(800),SINX(800),ANV1(800),
14 3/YPLN/  E0(800),E0(800),E0(800),E0(800),E0(800),E0(800),
15 4/TALWC/  EN(800),P(21,18,10),JL(21,18,10),16,MC(21,18,10),16,
16 1/CLAP/  AP(21,18,10),SU(21,18,10),SP(21,18,10),SU(21,18,10),
17 1/SPK(21,18,10),AE(21,18,10),AW(21,18,10),AN(21,18,10),
18 1/AS(21,18,10),AT(21,18,10),AB(21,18,10),APC(21,18,10),
19 19 00006I  COMMON
20 1/TRAN/  X(21,18,10),Y(21,18,10),Z(21,18,10),TJ0(21,18,10),
21 2/CK(21,18,10),CY(21,18,10),CZ(21,18,10),
22 3/EX(21,18,10),EY(21,18,10),EZ(21,18,10),
23 3/SW(21,18,10),SY(21,18,10),SZ(21,18,10),
24 1/LIMIT/  LIMIT1,LT,MT=1,LT,MT=1,LT,MT=1,LT,MT=1,LT,MT=1,LT,MT=1,
25 2/ALU/  ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,
26 00006I  COMMON
27 1/TRAN/  TXB(21,18,10),TXW(21,18,10),TYN(21,18,10),
28 2/TYM(21,18,10),TIZT(21,18,10),TIZB(21,18,10),
29 3/TYX(21,18,10),TYXW(21,18,10),TYZT(21,18,10),
30 4/TYS(21,18,10),TXYN(21,18,10),TXYS(21,18,10),
31 5/TXZ(21,18,10),TXZB(21,18,10),TIXE(21,18,10),
32 6/TXW(21,18,10),TYXW(21,18,10),TXYS(21,18,10),
33 1/UNSTY/  UNTV(21,18,10),XCV(21,18,10),MC(21,18,10),DK(21,18,10),DK(21,18,10),
34 2/CEQ(21,18,10),DENO(21,18,10),TD1(21,18,10),
35 00006I  LOGICAL INSU,INSOW,INSOP,INSOK,INSOL,INP,INSW,INSOS,INP
36 C**** INPUT DATA GUIDE *************
37 C
38 C NLIMIT: MAXIMUM NO. OF ITERATIONS LIMIT
39 C IG = 1: LAMINAR
40 C IG = 2: TURBULENT (X-E MODEL)
41 C ISWP: NO. OF SWEEPS FOR SOLVING THE P' EQUATION (PP)
42 C ITT: TOTAL NO. OF TIME STEPS
43 C REFL : REFERENCE REYNOLDS NUMBER
44 C ISWP: NO. OF SWEEPS FOR SOLVING THE P' EQUATION (PP)
45 C COTT: TIME STEP FOR UNSTEADY PROBLEMS
46 C C
47 C
48 C 00006I  REAC(5,100) NLIMIT,IG,ISWP,ITT
49 C 000030I  REAC(5,200) ALU,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP,ALP
50 C 00006C  REAC(5,200) REFL,ISWP
C-----CONSTANTS
58 000090I  5E=1.0E=3
59 000091I  ISW=7
60 000092I  ISW=7
61 000093I  ISW=1
62 000094I  ISW=5
63 000095I  ISW=5
64 000096I  ISWE=5
65 000097I  DENT=1.0
66 000098I  VISCA=1.0
67 000099I  SIGM=1.0
68 000100I  SIGM=1.0
69 000101I  SIGM=1.0
70 000102I  CMU=0.09
71 000103I  C1=1.43
72 000104I  C2=1.62
73 000105I  E=8.10C69
74 000106I  CK=0.5
75 000107I  PI=1.5708
76 000108I  M1=7.0E30
77 000109I  SMNL=1.0E30
78 000110I  CMU1=CMU*0.25
79 000111I  CMU2=CMU*0.75
80 000112I  INSCV=TRUE.
81 000113I  INSCP=TRUE.
82 000114I  INSCK=TRUE.
83 000115I  INSEC=TRUE.
84 000116I  INSTRF=TRUE.
85 000117I  INSTF=TRUE.
86 000118I  INSCHM=FALSE.
87 000119I  INSCHN=FALSE.
88 000120I  IPSQ=1.0
89 000121I  INSR=FALSE.
90 000122I  INSRO=FALSE.
91 000123I  INFSR=FALSE.
92 000124I  INSF=FALSE.
93 000125I  CONTINUE
94 000126I  C----READ IN INITIAL FLOW FIELDS FROM RESTART FILE (LU = 8)********
95 000127I  XI=0.1000 L/N/L1/L2/M1/M2/N1/N2
96 000128I  L=1
97 000129I  D=1
98 000130I  N0=1
99 000131I  M=1
100 000132I  NT=1
101 000133I  C----INITIALIZE VARIABLES
102 000134I  CALL INIT
103 000135I  C----RESTART FILE
104 000136I  CALL RESTART
105 000137I  DO 50 K=1,N
106 000138I  DO 50 I=1,L
107 000139I  CALL REC(8,400)
108 000140I  DO 50 J=1,M
109 000141I  CALL REC(8,500) X(I,J,K),Y(I,J,K),Z(I,J,K),U(I,J,K),V(I,J,K),
110  2 VISER=PENTH
111 000142I  U(I,J,K)=W(I,J,K)
112 000143I  V(I,J,K)=W(I,J,K)
113 000144I  W(I,J,K)=W(I,J,K)
114 000145I  TM(I,J,K)=TM(I,J,K)
115 000146I  GO TO 50
DO(i,j,k)=DK(i,j,k)
DO(i,j,k)=OE(i,j,k)
DENC(i,j,k)=DENC(i,j,k)
GO TO 121
C------------GET BOUNDARY CONTROL PARAMETERS
C------------SET BOUNDARY TURBULENCE QUANTITIES TO ZERO
DO 121 I=1,L
DO 121 K=1,N
IF(MC(I,J,K).LE.0.0) GC TO 122
GO TO 121
122 DK(I,J,K)=0.0
OE(I,J,K)=0.0
UC(I,J,K)=0.0
VX(I,J,K)=0.0
VY(I,J,K)=0.0
VZ(I,J,K)=0.0
C------------CALCULATE GRID TRANSFORMATION COEFFICIENTS
C------------TURBULENT VISCOITY
IF(INPRC) CALL NEWVIS
C------------CALCULATE INLET MASS FLOW RATE
FLOW=1.0
DO 900 I=1
DO 900 J=2,M
DO 900 K=2,N
UC=(U(I,J,K)+U(I,J-1,K)+U(I,J,K-1)+U(I,J-1,K-1))/4
DENC=DENC(1,J,K)+DENC(I,J-1,K)+DENC(I,J,K-1)+DENC(I,J-1,K-1))/4
PI1=PI1(I,J,K)+PI1(I,J-1,K)+PI1(I,J,K-1)+PI1(I,J-1,K-1))/4
PI2=PI2(I,J,K)+PI2(I,J-1,K)+PI2(I,J,K-1)+PI2(I,J-1,K-1))/4
PI3=PI3(I,J,K)+PI3(I,J-1,K)+PI3(I,J,K-1)+PI3(I,J-1,K-1))/4
C1=C1(I,J,K)+C1(I,J-1,K)+C1(I,J,K-1)+C1(I,J-1,K-1))/4
C3=C3(I,J,K)+C3(I,J-1,K)+C3(I,J,K-1)+C3(I,J-1,K-1))/4
AREA=SQRT(P1*P2*P3*P4)*SQRT(C1*C2*C3*C4)
FLOW=FLOW+C1*AREA*UC
C------------TRANSIENT PROCESS
1 CONTINUE
IT=1
C------------SOLUTION PROCEDURES START
1 CONTINUE
CALL SYMOUT(3,1/2,L2,M2,N)
ITR=1
C------------CONVERGENCE CHECK
WRITE(6,300) IT,ERRU,ERRV,ERRM,ERRK,EREC,U(7,2,6)
172 00136C1  ERRMAX=ERRM+ERV+ERV+ERRW        172 114
173 00136C1  IF(ITER.GE.2C .AND. ERRMAX.GT.1.ED) GO TO 99   173 115
174 00136C1  IF(ITER.GE.NLIMT .OR. ERRMAX.LE. EREXT) GO TO 99   174 117
175 00136C1  ITER=ITER+1            175 119
176 00136C1  GO TO 1              176 120
177  C-----PRINT CTLS SOLUTIONS             177
178 00136C1  CONTINUE       178 121
179 00136C1  WRITE(7,100) L,H,N,L1,L2,M1,M2,N1,N2       179 122
180 00136C1  CO 601 K=1,N                  180 123
181 00140C1  CO 601 I=1,L              181 124
182 00140C1  WRITE(7,400)            182 125
183 00140C1  CO 602 J=1,M            183 126
184 00140C1  XV=X(I,J,K)         184 127
185 00140C1  XV=Y(I,J,K)         185 128
186 00140C1  XV=Z(I,J,K)         186 129
188                                                 187 130
189 00160C1  CONTINUE        188 131
190 00170C1  CONTINUE        190 132
191 00170C1  CONTINUE        191 133
192 00170C1  WRITE(7,300) ITG,TMT        192 134
193 00170C1  IF(ITG.GT.OTT .OR. OLT .EQ. 0.0) GO TO 999        193 135
194 00170C1  ITG=ITG+1        194 136
195 00170C1  GO TO 2        195 138
196 00170C1  CONTINUE        196 139
197 00170C1  CONTINUE        197 140
198 00170C1  FORMAT(11F7.4)        198
199 00170C1  FORMAT(1X,15E10.2)        199
200 00170C1  FORMAT(1X)        200 200
201 00170C1  FORMAT(3F5.4,3E12.4,2X,6E11.4)        201 200
202 00170C1  STOP        202 140
203 00170C1  END        203 141

NO ERRORS:770 905-01JC MAINPROG CNS3C 02/21/86 09:45:56 TABLE SPACE: 11 KB
STATEMENT BUFFER: 20 LINES/1321 BYTES STACK SPACE: 181 WORDS
SINGLE PRECISION FLOATING PT SUPPORT REQUIRED FOR EXECUTION
SUBROUTINE DIRCCS

COMMON

1 /TURN, SIGK, SIGE, CMU, C2, CMU2, S, CK, MINUM, SMNUM, ANV1(800),
2 /YN(800), YNB(800), SINSX(800), SINSX(800), SINSX(800), SINSX(800), ANW1(800),
3 YPLN(800), TAUN(800), IBC(800), JBC(800), KBC(800), IITY(800),
4 TAUM(800), GEN(21, 18, 10), MC(21, 18, 10), MLJ(21, 18, 10), JITO,
5 1 /TRANS, X(21, 18, 10), Y(21, 18, 10), Z(21, 18, 10), TJO(21, 18, 10),
6 2 /CX(21, 18, 10), CY(21, 18, 10), CZ(21, 18, 10),
7 3 EX(21, 18, 10), EY(21, 18, 10), EZ(21, 18, 10),
8 4 SX(21, 18, 10), SY(21, 18, 10), SZ(21, 18, 10),
9 1 /LIPT, LM, LT, L1, L2, M1, M2, L2, M2, L1, M1, L2, M2, L1, M1, L2, M2,
10 /ANL1, ANL2, ALVs, ALN, ALV1, ALN1, ALV2, ALN2, ALV3, ALN3, ALV4,
11 /ADD, ALV5, ALV6, ALN4, ALV7, ALN5, ALV8, ALN6, ALV9, ALN7, ALV10,
12 /SUN, SUN1, SUN2, SUN3, SUN4, SUN5, SUN6, SUN7, SUN8, SUN9,
13 /GNS, GN1, GN2, GN3, GN4, GN5, GN6, GN7, GN8, GN9,
14 /SC, SC1, SC2, SC3, SC4, SC5, SC6, SC7, SC8, SC9,
15 /NRS, NRS1, NRS2, NRS3, NRS4, NRS5, NRS6, NRS7, NRS8, NRS9,
16 /PAR, PAR1, PAR2, PAR3, PAR4, PAR5, PAR6, PAR7, PAR8, PAR9,
17 /RPS, RPS1, RPS2, RPS3, RPS4, RPS5, RPS6, RPS7, RPS8, RPS9,
18 /RPLS, RPLS1, RPLS2, RPLS3, RPLS4, RPLS5, RPLS6, RPLS7, RPLS8, RPLS9,
19 /RPLS1, RPLS11, RPLS12, RPLS13, RPLS14, RPLS15, RPLS16, RPLS17, RPLS18, RPLS19,
20 /RPLS2, RPLS21, RPLS22, RPLS23, RPLS24, RPLS25, RPLS26, RPLS27, RPLS28, RPLS29,
21 /RPLS3, RPLS31, RPLS32, RPLS33, RPLS34, RPLS35, RPLS36, RPLS37, RPLS38, RPLS39,
22 /RPLS4, RPLS41, RPLS42, RPLS43, RPLS44, RPLS45, RPLS46, RPLS47, RPLS48, RPLS49,
23 /RPLS5, RPLS51, RPLS52, RPLS53, RPLS54, RPLS55, RPLS56, RPLS57, RPLS58, RPLS59,
24 /RPLS6, RPLS61, RPLS62, RPLS63, RPLS64, RPLS65, RPLS66, RPLS67, RPLS68, RPLS69,
25 /RPLS7, RPLS71, RPLS72, RPLS73, RPLS74, RPLS75, RPLS76, RPLS77, RPLS78, RPLS79,
26 /RPLS8, RPLS81, RPLS82, RPLS83, RPLS84, RPLS85, RPLS86, RPLS87, RPLS88, RPLS89,
27 /RPLS9, RPLS91, RPLS92, RPLS93, RPLS94, RPLS95, RPLS96, RPLS97, RPLS98, RPLS99,
28 /RPLS10, RPLS11, RPLS12, RPLS13, RPLS14, RPLS15, RPLS16, RPLS17, RPLS18, RPLS19,
29 /RPLS20, RPLS21, RPLS22, RPLS23, RPLS24, RPLS25, RPLS26, RPLS27, RPLS28, RPLS29,
30 /RPLS30, RPLS31, RPLS32, RPLS33, RPLS34, RPLS35, RPLS36, RPLS37, RPLS38, RPLS39,
31 /RPLS40, RPLS41, RPLS42, RPLS43, RPLS44, RPLS45, RPLS46, RPLS47, RPLS48, RPLS49,
32 /RPLS50, RPLS51, RPLS52, RPLS53, RPLS54, RPLS55, RPLS56, RPLS57, RPLS58, RPLS59,
33 /RPLS60, RPLS61, RPLS62, RPLS63, RPLS64, RPLS65, RPLS66, RPLS67, RPLS68, RPLS69,
34 /RPLS70, RPLS71, RPLS72, RPLS73, RPLS74, RPLS75, RPLS76, RPLS77, RPLS78, RPLS79,
35 /RPLS80, RPLS81, RPLS82, RPLS83, RPLS84, RPLS85, RPLS86, RPLS87, RPLS88, RPLS89,
36 /RPLS90, RPLS91, RPLS92, RPLS93, RPLS94, RPLS95, RPLS96, RPLS97, RPLS98, RPLS99,
37 /RPLS100, RPLS11, RPLS12, RPLS13, RPLS14, RPLS15, RPLS16, RPLS17, RPLS18, RPLS19,
38 /RPLS200, RPLS21, RPLS22, RPLS23, RPLS24, RPLS25, RPLS26, RPLS27, RPLS28, RPLS29,
39 /RPLS300, RPLS31, RPLS32, RPLS33, RPLS34, RPLS35, RPLS36, RPLS37, RPLS38, RPLS39,
40 /RPLS400, RPLS41, RPLS42, RPLS43, RPLS44, RPLS45, RPLS46, RPLS47, RPLS48, RPLS49,
41 /RPLS500, RPLS51, RPLS52, RPLS53, RPLS54, RPLS55, RPLS56, RPLS57, RPLS58, RPLS59,
42 /RPLS600, RPLS61, RPLS62, RPLS63, RPLS64, RPLS65, RPLS66, RPLS67, RPLS68, RPLS69,
43 /RPLS700, RPLS71, RPLS72, RPLS73, RPLS74, RPLS75, RPLS76, RPLS77, RPLS78, RPLS79,
44 /RPLS800, RPLS81, RPLS82, RPLS83, RPLS84, RPLS85, RPLS86, RPLS87, RPLS88, RPLS89,
45 /RPLS900, RPLS91, RPLS92, RPLS93, RPLS94, RPLS95, RPLS96, RPLS97, RPLS98, RPLS99,
46 /RPLS1000, RPLS11, RPLS12, RPLS13, RPLS14, RPLS15, RPLS16, RPLS17, RPLS18, RPLS19,
SUBROUTINE NEWVIS
COMMON
1/ VAR/U(21,19,10),V(21,19,10),P(21,18,10),DK(21,18,10),
2/ DE/21,18,10)/ERRU,ERRV,ERRM,ERRX,ERRY,ERRW,
3/ PP(21,18,10)/W(21,18,10)/TM/(21,18,10)
1/ PRCF/ VISE(21,18,10), CEN(21,18,10), VISC, DENIX, FLOWIN
1/ PCCF/ CU(21,18,10),OY(21,18,10),OM(21,18,10)
1/ TUR/ SIGK/SIGE/CMU/C1/C2/CMU2/E/CMK/SNUNUM/SMNU/ANV1(800),
2/ YN(800)/YN(800)/SINX(800)/SINX(800)/SINZ(800)/SINZ(800)/ANW1(800)
3/ YPLN(SCD)/TUM(BCD)/ISC(BCD)/AIC(BCD)/AIC(BCD)/ANV(Y) I/TV/600)
4/ TALW(800)/GEN(21,13,10)/MC(21,18,10)/I/JL0(21,18,10)/I/I01
COMMON
1/ LINFY LP/LLT/L1/L12/W1/W2/LQ/M0/ISWU/ISWW/ISWP/ISWK/ISWE/
2/ AL/ALP/ALK/ALW/ALV/ALW/N1/N2/N3/ISW/ISW/ISW/ICNT/ALC/DTT
C-----EVALUATE TURBULENT VISCOSITY
CO 10 I=1,L
CO 10 J=1,K
IF(CI(I,J,K),LE.,SMNU)=SMNU
IF(CI(I,J,K),LE.,SMNU)=SMNU
IF(CI(I,J,K),LE.,SMNU)=SMNU
GO TO 12
TURVIS=CEN(I,J,K)*CMU*CK(I,J,K)**2/DE(I,J,K)*VISC
GO TO 14
TURVIS=VISC
CONTINUE
VIS(I,J,K)=VISION(I,J,K)*ALVIS*(TURVIS-VISE(I,J,K))
CONTINUE
RETURN
END

ERRORS=170 005-01.CC SUBROUTINE NEWVIS 02/21/86 09:51:23 TABLE SPACE: 5 KB
STATEMENT BUFFER: 20 LINES/1321 BYTES STACK SPACE: 203 WORDS
SINGLE PRECISION FLOATING PT SUPPORT REQUIRED FOR EXECUTION
C-----PRESSURE AND VELOCITIES CORRECTIONS

DO 880 I=IS,IT
DO 880 J=JL,JT
DO 880 K=KS,KT
IF(KC(I,J,K).GE.2) GC TO 880
P=FC(I,J,K)-PPREF
F(I,J,K)=F(I,J,K)+ALM*P
ERRF=AMX1(ERRF,ABS(P))
CONTINUE
880 CONTINUE

DO 880 I=2,LT
DO 880 J=2,MT
DO 880 K=2,NT
IF(KC(I,J,K).GE.1) GC TO 600
I=I+1
J=J+1
K=K+1
PE=(F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K))*0.25
P=FC(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K)*0.25
PN=(F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K))*0.25
PS=(F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K))*0.25
PT=(F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K))*0.25
PS=(F(I,J,K)+F(I,J,K)+F(I,J,K)+F(I,J,K))
PCXI=PE-PN
PDEC=PN-PS
PCXI=CT(I,J,K)
EXE=EX(I,J,K)
SX=EX(I,J,K)
CYC=CYC(I,J,K)
ECY=ECY(I,J,K)
SAX=SC(I,J,K)
CZ=CZ(I,J,K)
ZC=ZC(I,J,K)
SIZ=ES(I,J,K)
SIG=SI(I,J,K)
PAX=PCXI*CQ*PDEC*EXC*PSCI*SXQ
PYY=PCXI*CQ*PECA*EQC*PSCI*SYQ
PZI=PCXI*CQ*PSCI*EQC*PSCI*SEQ
UC(I,J,K)=UC(I,J,K)*CP(I,J,K)*PXX
VCT=VCT(I,J,K)+DUCD(I,J,K)*PYY
WCT=WCT(I,J,K)+DUCD(I,J,K)*PZZ
CONTINUE
600 CONTINUE
RETURN
END

NO ERRORS:7FD ROI=01.0C SUBROUTINE Solved 02/21/86 09:58:31 TABLE SPACE: 18 KB
STATEMENT BUFFER: 20 LINES/132 BYTES STACK SPACE: 203 WORDS
SINGLE PRECISION FLOATING PT SUPPORT REQUIRED FOR EXECUTION
SUBROUTINE LINERS(ISL,IS,J,S,K,LT,MT,NT,F)

DIMENSION A(20),B(20),C(20),D(20),F(21,18),T(21,18,10)

COMMON /C,F/ A(21,18,10),B(21,18,10),C(21,18,10),D(21,18,10)

C-------- LINEAR EQUATIONS SOLVERS

GO TO (1) ISL

1 C-------- LINE-RELAXATION USING TMA

CONTINUE

GO 10 I=IS,LT

GO 10 J=JS,MT

GO 10 K=KS,NT

1 BS(I,J,K)=AE(I,J,K)*F(I-1,J,K)+AN(I,J,K)*F(I,J+1,K)+

SU(I,J,K)

10 CONTINUE

13 EE(I,J,K)=0,L,C

DO 100 I=IS,LT

DO 100 J=JS,MT

DO 100 K=KS,NT

14 EE(I,J,K)=AT(I,J,K)

15 EE(I,J,K)=AB(I,J,K)

16 EE(I,J,K)=BE(I,J,K)*AW(I,J,K)+F(I-1,J,K)+

17 EE(I,J,K)=AS(I,J,K)*F(I,J-1,K)

18 EE(I,J,K)=AK(I,K)*F(I,J,K)+1.E-30

19 EE(I,J,K)=T(I,J,K)

20 EE(I,J,K)=TERM(I,J,K)

21 EE(I,J,K)=TERM(I,J,K)

22 EE(I,J,K)=TERM(I,J,K)

23 EE(I,J,K)=TERM(I,J,K)

24 EE(I,J,K)=TERM(I,J,K)

25 EE(I,J,K)=TERM(I,J,K)

26 EE(I,J,K)=TERM(I,J,K)

27 EE(I,J,K)=TERM(I,J,K)

28 EE(I,J,K)=TERM(I,J,K)

29 EE(I,J,K)=TERM(I,J,K)

30 EE(I,J,K)=TERM(I,J,K)

31 EE(I,J,K)=TERM(I,J,K)

32 EE(I,J,K)=TERM(I,J,K)

33 EE(I,J,K)=TERM(I,J,K)

34 EE(I,J,K)=TERM(I,J,K)

35 EE(I,J,K)=TERM(I,J,K)

36 EE(I,J,K)=TERM(I,J,K)

37 EE(I,J,K)=TERM(I,J,K)

38 EE(I,J,K)=TERM(I,J,K)

39 EE(I,J,K)=TERM(I,J,K)

40 EE(I,J,K)=TERM(I,J,K)

41 EE(I,J,K)=TERM(I,J,K)

42 EE(I,J,K)=TERM(I,J,K)

43 EE(I,J,K)=TERM(I,J,K)

44 EE(I,J,K)=TERM(I,J,K)

45 EE(I,J,K)=TERM(I,J,K)

46 EE(I,J,K)=TERM(I,J,K)

47 EE(I,J,K)=TERM(I,J,K)

48 EE(I,J,K)=TERM(I,J,K)

49 EE(I,J,K)=TERM(I,J,K)

50 EE(I,J,K)=TERM(I,J,K)

51 EE(I,J,K)=TERM(I,J,K)

52 EE(I,J,K)=TERM(I,J,K)

53 EE(I,J,K)=TERM(I,J,K)

54 EE(I,J,K)=TERM(I,J,K)

55 EE(I,J,K)=TERM(I,J,K)

56 EE(I,J,K)=TERM(I,J,K)

57 EE(I,J,K)=TERM(I,J,K)

58 EE(I,J,K)=TERM(I,J,K)

59 EE(I,J,K)=TERM(I,J,K)

60 EE(I,J,K)=TERM(I,J,K)

61 EE(I,J,K)=TERM(I,J,K)

62 EE(I,J,K)=TERM(I,J,K)

63 EE(I,J,K)=TERM(I,J,K)

64 EE(I,J,K)=TERM(I,J,K)

65 EE(I,J,K)=TERM(I,J,K)

66 EE(I,J,K)=TERM(I,J,K)

67 EE(I,J,K)=TERM(I,J,K)

68 EE(I,J,K)=TERM(I,J,K)

69 EE(I,J,K)=TERM(I,J,K)

70 EE(I,J,K)=TERM(I,J,K)

71 EE(I,J,K)=TERM(I,J,K)

72 EE(I,J,K)=TERM(I,J,K)

73 EE(I,J,K)=TERM(I,J,K)

74 EE(I,J,K)=TERM(I,J,K)

75 EE(I,J,K)=TERM(I,J,K)

76 EE(I,J,K)=TERM(I,J,K)

77 EE(I,J,K)=TERM(I,J,K)

78 EE(I,J,K)=TERM(I,J,K)

79 EE(I,J,K)=TERM(I,J,K)

80 EE(I,J,K)=TERM(I,J,K)

81 EE(I,J,K)=TERM(I,J,K)

82 EE(I,J,K)=TERM(I,J,K)

83 EE(I,J,K)=TERM(I,J,K)

84 EE(I,J,K)=TERM(I,J,K)

85 EE(I,J,K)=TERM(I,J,K)

86 EE(I,J,K)=TERM(I,J,K)

87 EE(I,J,K)=TERM(I,J,K)

88 EE(I,J,K)=TERM(I,J,K)

89 EE(I,J,K)=TERM(I,J,K)

90 EE(I,J,K)=TERM(I,J,K)

91 EE(I,J,K)=TERM(I,J,K)

92 EE(I,J,K)=TERM(I,J,K)

93 EE(I,J,K)=TERM(I,J,K)

94 EE(I,J,K)=TERM(I,J,K)

95 EE(I,J,K)=TERM(I,J,K)

96 EE(I,J,K)=TERM(I,J,K)

97 EE(I,J,K)=TERM(I,J,K)

98 EE(I,J,K)=TERM(I,J,K)

99 EE(I,J,K)=TERM(I,J,K)

100 EE(I,J,K)=TERM(I,J,K)
ORIGINAL PAGE IS OF POOR QUALITY
58 000441  
59 000461  
60 000481  60 CONTINUE  
61 0004a1  RETURN  
62 C------ LINK COEFF. MODIFICATIONS  
63 000511  2 CONTINUE  
64 C------ EAST OUT  
65 000541  I=IT  
66 000561  DO 200 J=2, JT  
67 000581  DO 200 K=2, KT  
68 0005a1  AE(I,J,K)=0.0  
69 0005c1  200 CONTINUE  
70 0005e1  RETURN  
71 C------ UPDATE LNSTEADY COEFF.  
72 000611  3 CONTINUE  
73 000631  IF(I,J,K, NE, 0.0) GO TO 301  
74 000651  DO 300 I=1, IT  
75 000671  DO 300 J=1, JT  
76 000691  DO 300 K=1, KT  
77 0006b1  300 APQ(I,J,K)=0.0  
78 0006d1  RETURN  
79 0006f1  301 CONTINUE  
80 000711  DO 310 I=1, IT  
81 -000731  DO 310 J=1, JT  
82 000751  DO 310 K=1, KT  
83 000771  APO(I,J,K)=300(I,J,K)/DTT  
84 000791  UO(I,J,K)=U(I,J,K)  
85 0007b1  VO(I,J,K)=V(I,J,K)  
86 0007d1  WQ(I,J,K)=W(I,J,K)  
87 0007f1  TMG(I,J,K)=TM(I,J,K)  
88 000811  DGK(I,J,K)=DE(I,J,K)  
89 000831  DEGIJ,K)=DE(I,J,K)  
90 000851  310 CONTINUE  
91 000871  RETURN  
92 000891  END  

NO ERRORS; F70 ROS-01.0C SUBROUTINE SYMOLT 02/21/86 10:01:59 TABLE SPACE: 7 KB  
STATEMENT BUFFER: 20 LINES/1371 BYTES STACK SPACE: 131 WORDS  
SINGLE PRECISION FLOATING PT SUPPORT REQUIRED FOR EXECUTION
APPROVAL

A COMPUTER CODE FOR THREE-DIMENSIONAL INCOMPRESSIBLE FLOWS USING NONORTHOGONAL BODY-FITTED COORDINATE SYSTEMS

By Y. S. Chen

The information in this report has been reviewed for technical content. Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

G. F. McDonough
Director, Systems Dynamics Laboratory