View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

NASA Technical Memorandum 100012

INS3D–An Incompressible Navier-Stokes Code in Generalized Three-Dimensional Coordinates

S. E. Rogers, Sterling Federal Systems, Palo Alto, California
D. Kwak, Ames Research Center, Moffett Field, California
J. L. C. Chang, Rockwell International, Canoga Park, California

November 1987

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

INS3D – AN INCOMPRESSIBLE NAVIER-STOKES CODE IN GENERALIZED THREE-DIMENSIONAL COORDINATES

Stuart E. Rogers
Sterling Federal Systems
Palo Alto, California

Dochan Kwak
NASA Ames Research Center
Moffett Field, California

James L. C. Chang
Rocketdyne Division, Rockwell International
Canoga Park, California

SUMMARY

The operation of the INS3D code, which computes steady-state solutions to the incompressible Navier-Stokes equations, is described. The flow solver utilizes a pseudocompressibility approach combined with an approximate factorization scheme. This publication describes key operating features to orient new users. This includes the organization of the code, description of the input parameters, description of each subroutine, and sample problems. Details for more extended operations, including possible code modifications, are given in the Appendix.

INTRODUCTION

This publication is intended as a reference guide for users of the INS3D code (ref. 1). Many recent advances in computational fluid dynamics (CFD) have been made in conjunction with compressible flow computations. Therefore, the code INS3D is structured to take advantage of fast algorithms developed for compressible flows. The primary features can be summarized as follows.

(1) A pseudocompressibility approach is used in which a time derivative of pressure is added to the continuity equation, which together with the momentum equations form a set of four equations with pressure and velocity as the dependant variables (refs. 2, 3, 4).
(2) The coordinates are transformed for general three-dimensional applications.
(3) This set of equations is solved using an approximate-factorization scheme (refs. 5, 6).
(4) Computational efficiency is achieved using a diagonal algorithm (ref. 7). A block tridiagonal option is also available.
(5) When a steady-state solution is reached, the modified continuity equation will satisfy the divergence-free velocity field condition.

The code has been verified by computing fundamental fluid dynamics problems such as the flow through a channel (ref. 1), flow over a backward-facing step (ref. 8), and flow over a circular cylinder (refs. 1 and 8). Three-dimensional cases include flow over an ogive cylinder(ref. 9), flow through a rectangular duct(ref. 1), wind-tunnel-inlet flow (ref. 10), cylinder-wall juncture flow and flow through multiple posts mounted between two plates (refs. 11 and 12). The most striking application of the present code is the flow simulation in the power head portion of the Space Shuttle Main Engine (refs. 13 and 14).

GENERAL USER NOTES

This section contains some important features of INS3D which will be useful for the first-time user. The arrays in the program are dimensioned using a single index I, which is a function of the three single-direction indices.

$$\text{I} = \text{J} + (\text{K-1})*\text{JMAX} + (\text{L-1})*\text{JMAX}*\text{KMAX}$$

Here, J is the index in the ξ-direction, K is the index in the η-direction, and L is the index in the ζ-direction. The grid point locations are stored in the single index arrays X, Y, and Z. The primitive variables are stored in the array Q, where

$$\begin{aligned} \text{Q(1,I)} &= \text{pressure} \\ \text{Q(2,I)} &= \text{u velocity component} \\ \text{Q(3,I)} &= \text{v velocity component} \\ \text{Q(4,I)} &= \text{w velocity component} \end{aligned}$$

To run a specific geometry, the user will have to change a number of different subroutines. These subroutines are GRID, IC, BC, and SMOOTH. The details of each of these changes are included here.

The grid coordinates for the problem should be loaded into the X, Y, and Z arrays in subroutine GRID. This subroutine is called once at the beginning of the program execution. If the grid can be generated algebraically, this can be done in this subroutine. If the coordinates are generated with some other program, then the coordinates merely have to be read in and stored in the X, Y, and Z arrays. In defining the geometry for the problem, the coordinates should be normalized by the characteristic length of the problem because the program assumes a characteristic length of unity when it uses the input Reynolds number.

The subroutine IC should be changed to set the initial conditions for the the specific problem. It is usually adequate to set the pressure everywhere to unity, and to set the velocity components to freestream conditions everywhere except on no-slip walls, where zero velocity should be specified. The velocity field should also be normalized with the characteristic velocity for the problem because the program assumes a characteristic velocity of unity when using the Reynolds number.

The third subroutine which needs to be changed is BC, where the explicit boundary conditions are applied. By default, the code will update all the interior points during one iteration,

and then call subroutine BC. See the Numerical Algorithm section in the appendix for more details on the proper application of boundary conditions.

Changes may also have to be made to the subroutine SMOOTH, which adds the artificial dissipation to the right side of the equations. In general, fourth-order explicit smoothing is added at all the interior grid points. Stability of the calculations is enhanced if second-order smoothing is added at the interior points adjacent to the inflow and outflow boundaries. Sometimes when stability problems occur near severe geometries or grid areas, the use of second-order explicit smoothing will help to smooth out the disturbance.

Another change of primary importance is to correctly set the dimensions of the arrays for the specific problem. This is easily done with the use of one parameter statement which is included in almost all the subroutines. Two parameters are included in this statement, JKLMAX and IMX. The first parameter JKLMAX should be set to the maximum of JMAX, KMAX, or LMAX. The second parameter IMX should be set equal to the total number of grid points, that is, the product of JMAX, KMAX, and LMAX. All of the arrays are passed between the subroutines with the use of common blocks.

Periodic cases can be handled if the problem is set up so that the periodic direction corresponds with the η-direction. The K=KMAX and K=1 planes must be adjacent to each other and are treated as interior points in this case.

Two-dimensional problems can be handled by setting JMAX, KMAX, or LMAX to three. For instance, if the two-dimensional problem is chosen to lie in the JK plane in computational space, then LMAX should be set to three. Assuming that the physical space is set up to lie in the xy plane, the grid should be duplicated in three z = constant planes. The dependant variables should be initialized to have identical values in the L=1, 2, and 3 planes. No calculations will be done in the direction which has a maximum dimension of three, and calculations will only be done on the middle of the three planes. A loop should be put in subroutine BC which copies the data from the middle plane to the two outer planes after each iteration.

VARIABLES IN NAMELIST DATAIN

The input for the program is read in with the use of a namelist. Each of the variables in this namelist are described here. This is done in tabular form; the entries are the name of the variable, the possible range of acceptable values, its default value, and a description of the variable.

Variable	Range	Default	Description
BETA	0.1 to 50.0	5.0	This is the artificial compressibility parameter. The artificial speed of sound is a function of this parameter, and the minimum value this should take is dependent on the geometry. The maximum value is bounded because of the error introduced by the approximate factorization. For more details, see the pseudocompressibility section in the Appendix.
DISKOUT	True or False	False	This is a logical variable which controls whether subroutine DISKIO is called at the end of the program to write the Q array to disk. The array is written out to unit 51.
DTAU	0.0001 to 0.1	0.05	This is the time step used at each iteration. If a stability problem occurs, this parameter should be reduced.
DXDT	$-\infty$ to ∞	0.0	This is a parameter which is used to specify translation of the grid in the x-direction. It is used when the time varying metrics are computed.
ENDACC	0 or 1	1	This is an integer which controls the spatial differencing of the metrics at points next to nonperiodic boundaries. If ENDACC=1, first-order-accurate one-sided differencing is used. Otherwise, second-order-accurate one-sided differencing is used.
IBLKDIA	1 or 2	2	This integer determines whether the code will use a standard block algorithm for the inversion of the approximately factored matrices or a diagonalized algorithm. For IBLKDIA = 1, the standard algorithm is used; for IBLKDIA = 2, the diagonal algorithm is used. The standard algorithm inverts 4-by-4 block tridiagonal matrices whereas the diagonal algorithm inverts scalar tridiagonal matrices. The diagonal algorithm is about three times faster for a single time step than the standard algorithm, and has the same convergence characteristics.

Variable	Range	Default	Description
IMPSMO	2 or 4	2	This integer determines the order of the implicit artificial dissipation when the diagonal algorithm is used. When set equal to 2, second-order dissipation is used, when set equal to 4, fourth-order dissipation is used. Not much difference has been seen between the two different orders. The fourth order is slightly more expensive as it requires the inversion of scalar pentadiagonal matricies.
IORTHO	1 or 2	2	This integer affects the assumptations in computing the viscous terms. For IORTHO=1, the nonorthogonal contributions to the viscous terms are included in the right-hand side of the equations by calling subroutine VISRHS2. For IORTHO=2, the viscous terms are simplified by assuming an orthogonal grid.
IPRNT	1 to NTMAX	1	This integer controls the frequency with which the convergence history is written. Convergence data is written every IPRNTth iteration.
ISTART	0 or 1	0	This variable is an integer which controls the restart capability of the code. If ISTART = 0, then the code starts from the conditions specified in subroutine IC. If ISTART = 1, then the code calls subroutine DISKIO which attempts to read in NT, JMAX, KMAX, LMAX and the Q array from unit 20. The grid is always computed in subroutine GRID regardless of the value of ISTART.
JMAX	≥ 3	52	This integer is the maximum number of points used in the ξ-direction.
KMAX	≥ 3	63	This integer is the maximum number of points used in the η-direction.
KPERI	0 or 1	1	This integer allows one to specify a periodic boundary between the K=KMAX and K=1 planes. If KPERI = 1, the boundary is assumed to be periodic and the points at K=1 and K=KMAX are treated as interior points. If KPERI = 0, these points are treated as normal boundary points.
LMAX	≥ 3	35	This integer is the maximum number of points used in the ζ-direction.

Variable	Range	Default	Description
NPRNT	≥ 1	50	This integer controls the frequency with which flow solution output is to be written during the iteration process. Every NPRNTth iteration subroutine OUTPUT is called, which prints out the values for Q, X, Y, Z, Cp, and the Jacobian for specified indices.
NTMAX	≥ 1	100	This integer is the number of iterations the program will run.
PLOT3D	True or False	False	This logical variable determines whether unformatted files to be used with the plotting program known as PLOT3D will be written at the end of the computation. If the variable is TRUE, the grid file is written to unit 30, and the Q file is written out to unit 31.
REYNUM	0.0	1000	This is the Reynolds number based on unit characteristic length and unit characteristic velocity.
RL1	0.0 or 1.0	1.0	This variable is used for a geometry in which the volume of the grid cells becomes zero at L = 1. This can occur with an O-grid whose innermost circle has a radius of zero. This variable then represents the radius at the L = 1 circle, and is used only when it is set equal to zero.
SMU	0.0	0.1	This real number is the smoothing coefficient used for the explicit dissipation added to the right-hand side. This should be one-half to one-third the value of SMUIM.
SMUIM	0.0	0.3	This real number is the smoothing coefficient used for the implicit dissipation added to the left-hand side. This should be two to three times the value of SMU.
SMUPRS	0.0	1.0	This real number is a coefficient which multiplies the smoothing terms added to the right-hand side of the pressure equation in addition to SMU. It may become necessary to reduce this coefficient as the solution converges so that the divergence of the velocity will continue to converge toward zero. For more details, see the section on the effects of smoothing on pressure in the pseudocompressibility section of the Appendix.

DESCRIPTION OF ROUTINES AND CODE STRUCTURE

In this section, a brief description of the subroutines found in INS3D is given. A flow chart of the main code and of subroutine STEP is then presented.

Program MAIN

Program MAIN handles the main control of the program. It first calls routine INITIA and then enters the main time-iteration loop. Inside this loop, the routines STEP, BC, and VISCT are called. If printing of the solution is requested for the current iteration, then OUTPUT is also called. After the iteration process is finished, MAIN also handles the writing of the solution to disk by calling DISKIO and P3DOUT if requested.

Subroutine AMATRX

Subroutine AMATRX computes the Jacobian matrix $\hat{A}_i$ and loads it into the D array in common block DUDD. It is computed and stored for all the points along a single grid line (two of the three j,k,l indicies held constant). The proper metrics must be loaded into the metric storage array depending on whether $\hat{A}, \hat{B}, \text{or } \hat{C}$ is being computed. The Jacobian matrix is used to fill the implicit side matrices.

Subroutine BC

Subroutine BC enforces the boundary conditions explicitly. It is called at the end of each iteration, after all the interior points have been updated to the new time-step. This routine must be changed for each specific geometry run.

Subroutine BTRI4 and PBTRI4

Subroutines BTRI4 and PBTRI4 invert a system of block tridiagonal matrices where the block size is 4 by 4. The system of equations must be loaded into the matrices in the BTRI common block. The solution is overwritten onto the forcing-function array F. BTRI4 handles the nonperiodic case and PBTRI4 handles the periodic case.

Subroutine COMET

Subroutine COMET computes and stores the elements of the first two eigenvectors of the Jacobian matrix. These quantities are used in computing the similarity transformation which diagonalizes the Jacobian matrix. This routine must be called prior to calling TK or TKINV.

Subroutine DISKIO

Subroutine DISKIO is used to read or write the flow field solution (the array) from the disk. Data is read from fortran unit 20, and is written to fortran unit 51.

Subroutine ETAINV, XIINV, and ZETAINV

Subroutines ETAINV, XIINV, and ZETAINV load the implicit side matrices for each of the three different sweeps of the diagonal algorithm. The eigenvalues of the Jacobian matrix are computed and differenced to form the elements of the diagonal matrices. The implicit viscous terms are added and then either second- or fourth-order artificial dissipation terms are added. The system is then inverted by the appropriate scalar inversion routines. Since the first two eigenvalues are identical, the left-hand sides of the first two equations are identical; only the forcing function is different. Therefore these two are inverted together, and the last two equations are inverted separately.

Subroutine FILTRJ, FILTRK, FILTRL, AND PFILTK

Subroutines FILTRJ, FILTRK, FILTRL, and PFILTK load the implicit-side matrices for each of the three different sweeps of the block algorithm. PFILTK is used for the periodic case. These routines load the matrices with the difference of the Jacobian matrices and add the implicit viscous terms as well as second-order artificial dissipation terms. The system is then inverted using BTRI4 or PBTRI4.

Subroutine GRID

Subroutine GRID is used to compute the computational grid and load the X, Y, and Z arrays. GRID is called at the beginning of every run whether or not it is a restart. This routine must be changed for each specific geometry run.

Subroutine IC

Subroutine IC is called at the beginning of all nonrestart runs and is used to initialize the flow-field-array Q and the eddy-viscosity-array VNUT. The desired starting conditions must be coded for each specific geometry run.

Subroutine INITIA

Subroutine INITIA initializes some variables and arrays and reads in the input namelist to obtain the user-specified parameters. It calls routines GRID, JACOB, and then calls DISKIO if the run is a restart or IC if it is not.

Subroutine JACOB

Subroutine JACOB computes the Jacobian of the transformation for the given grid and stores it in array DJ.

Subroutine LUDEC and LUSOL

Subroutines LUDEC and LUSOL perform L-U decompostions and solutions for a 4-by-4 matrix. These subroutines are called by subroutines BTRI4 and PBTRI4.

Subroutine METRIC

Subroutine METRIC computes the metrics for a constant grid line and stores them in the metric arrays in common block MET1.

Subroutine OUTPUT

Subroutine OUTPUT prints out the flow variables, the x, y, and z location, the Jacobian and the pressure coefficient for points along a given line.

Subroutine P3DOUT

Subroutine P3DOUT writes out the flow solution and the grid point variables in a format compatible with the PLOT3D plotting program. The grid-file is written to Fortran unit 30 and the q-file is written out to Fortran unit 31.

Subroutine PENTA, PENTA2, PENTAP, and PENTAP2

Subroutines PENTA, PENTA2, PENTAP, and PENTAP2 invert a scalar pentadiagonal system of equations. A whole plane of equations are inverted at once and must be loaded into the arrays in the PNTD common block. A whole plane of data is done at once so that the loops can be fully vectorized in the nonsweep direction. These routines are called by the diagonal algorithm routines when fourth-order implicit artificial dissipation is used.

Subroutine RHS

Subroutine RHS computes the right-hand side of the governing equations by computing the spatial difference of the flux vectors at each point. Artificial dissipation is added by calling subroutine SMOOTH and the right-hand side viscous terms are added by calling subroutine VISRHS and subroutine VISRHS2. The right-hand side of the equations is stored in array S.

Subroutine SMOOTH

Subroutine SMOOTH adds the explicit fourth-order artificial dissipation terms to the right-hand side in the S array for all interior points. At most of the points next to the boundaries, a shifted fourth-order stencil is used. At points next to the inflow and outflow boundaries, second-order dissipation should be used to help maintain stability. This routine should be changed for each new geometry run.

Subroutine STEP

Subroutine STEP marches the solution forward in time one iteration. It first calls RHS, and then successively calls three implicit-sweep routines. Depending on the value of IBLKDIA, either the block algorithm routines are called or the diagonal algorithm routines are used. After the inversion process is finished for all three sweeps, the flow variables at all interior points are updated to the new time-step.

Subroutine TK and TKINV

Subroutines TK and TKINV load the eigenvector matrix (TK) and its inverse (TKINV) and multiplies them by the forcing function stored in the S array. This is done for a single grid line at a time. The proper metrics must be loaded in the common block MBUF, and the elements of the first two eigenvectors must be loaded by the subroutine COMET. These routines are used in the diagonal inversion process.

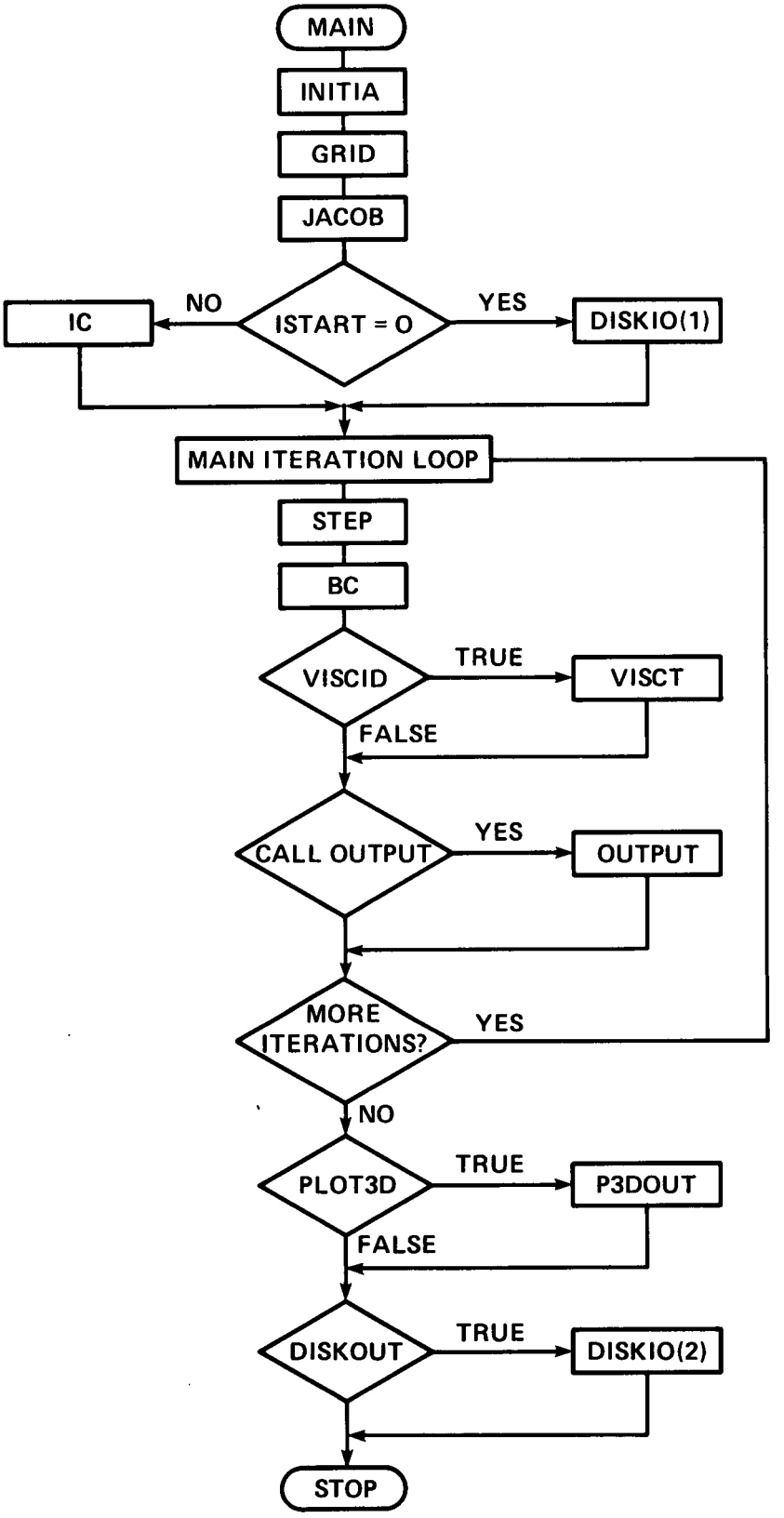
Subroutine TRI, TRI2, TRIP, and TRIP2

Subroutines TRI, TRI2, TRIP, and TRIP2 invert a scalar pentadiagonal system of equations. A whole plane of equations are inverted at once and must be loaded into the arrays in the PNTD common block. A whole plane of data is done at once so that the loops can be fully vectorized in the nonsweep direction. These routines are called by the diagonal algorithm routines when second-order implicit artificial dissipation is used.

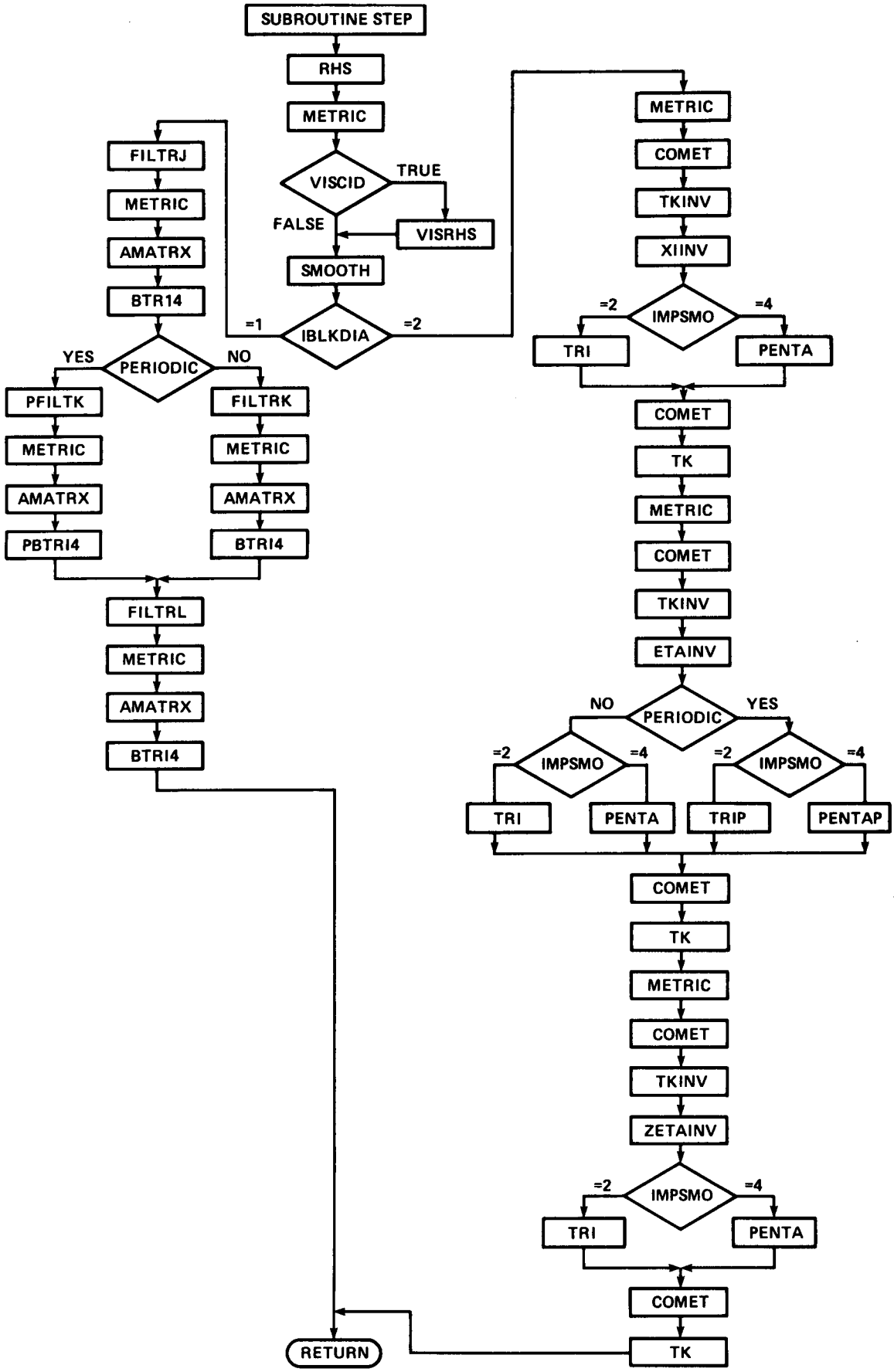
Subroutine VISRHS and VISRHS2

Subroutines VISRHS and VISRHS2 compute the right-hand-side viscous terms and adds them to the S array for all the interior points. The viscous terms for an orthogonal mesh are computed in VISRHS, and the nonorthogonal terms are computed in VISRHS2.

INS3D FLOWCHART



SUBROUTINE STEP FLOWCHART



SAMPLE CALCULATIONS

In this section three sample calculations are presented. The first example is a two-dimensional calculation of flow over a backward-facing step, and the second is the flow around a two-dimensional circular cylinder. The third case is flow around a three-dimensional post mounted normal and between two flat plates. Both the diagonal algorithm (IBLKDIA = 2) and the block algorithm (IBLKDIA = 1) were used for the backward-facing step, and the diagonal algorithm was used for the last two cases. The calculations were carried out on a Cray XMP-48. The average computer time on this machine used by the code for each case is approximately 3 x 10^{-5} secs per grid point per iteration for the diagonal algorithm. The block algorithm is between 2.5 and 3 times slower than this for most cases.

Backward-Facing Step

The flow over a backward-facing step with a two-to-one expansion ratio was computed for a laminar range of Reynolds numbers (based on twice the step height) between 100 and 800. A 65-by-33 algebraically-generated H-grid was used with the inflow boundary placed at the step. The grid was clustered near all the no-slip walls using an exponential stretching function.

The relevant parameters used in this calculation are given in the following list.

BETA	= 5	KPERI	= 0
DTAU	= 0.1	LMAX	= 65
ENDACC	= 1	SMU	= 0.1
IORTHO	= 2	SMUIM	= 0.3
JMAX	= 3	SMUPRS	= 0.1
KMAX	= 33		

The length of the separation bubble behind the step was calculated for each Reynolds number and these results are compared to the experimental results of Armaly et al. (ref. 15). This comparison is shown in Figure 1, which shows good agreement between the experiment and the calculations for both the diagonal algorithm and the block algorithm. The convergence history is shown in Figure 2. Here, the rms of the change in the flow variables (RMSDQ) at each time-step is plotted versus iteration number. Three cases are plotted: (1) the block algorithm, (2) the diagonal algorithm with second-order implicit smoothing, and (3) the diagonal algorithm with fourth-order implicit smoothing. The general convergence histories are the same for all three cases, and they are all able to converge about three-and-a-half orders of magnitude in less than 1000 iterations.

Circular Cylinder

The flow over a two-dimensional circular cylinder is a simple example of external flow over a bluff body which will have separation. Some of the flow quantites from the calculation were compared to experimental values in reference 1 and agreement was found.

An O-grid which wraps around the cylinder is used for the grid. The ξ- and η-directions

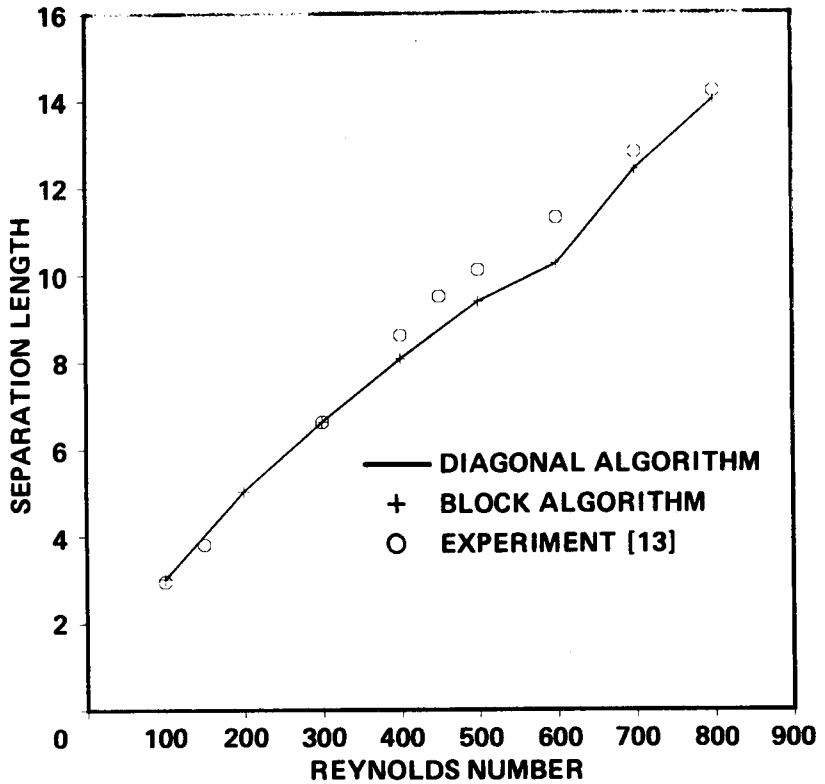


Figure 1.- Separation length versus Reynolds number for flow over a backward facing step.

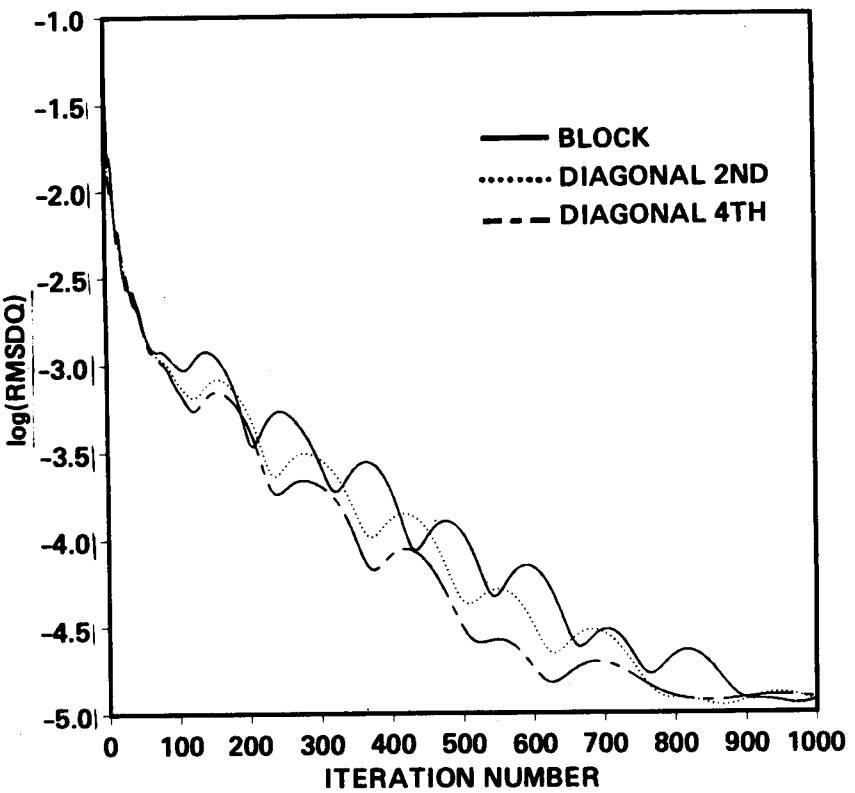


Figure 2.- Convergence history for flow over a backward facing step.

are shown in Figure 3. The ξ axis points in the radial direction and the η axis points along the circumferential direction since any periodic direction must be associated with the η direction.

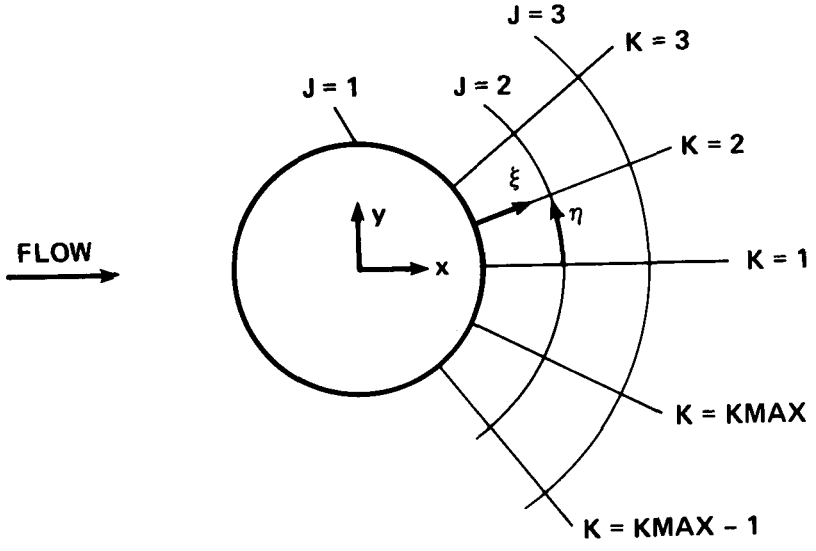


Figure 3.- Schematic diagram of an O-grid around a circular cylinder.

The relevant parameters used in this calculation are given in the following list.

BETA	= 5	KPERI	= 1
DTAU	= 0.1	LMAX	= 41
ENDACC	= 1	SMU	= 0.1
IORTHO	= 2	SMUIM	= 0.3
JMAX	= 3	SMUPRS	= 0.1
KMAX	= 80		

The convergence history is shown in Figure 4 in which the log of RMSDQ and the log of RMSDIV are plotted versus the iteration number. The variable RMSDQ is the rms of the change of all the variables during one iteration and RMSDIV is the rms of the divergence of velocity calculated during that time-step.

This calculation is typical of most external flow problems in that the calculation is not as sensitive to the downstream boundary conditions as in an internal flow problem. This calculation was found to work well when the downstream variables were updated using only a second-order extrapolation.

Three-Dimensional Post

The geometry of this calculation consists of a circular post mounted between two parallel flat plates, such that the post is perpendicular to the plates and the incoming flow is parallel to

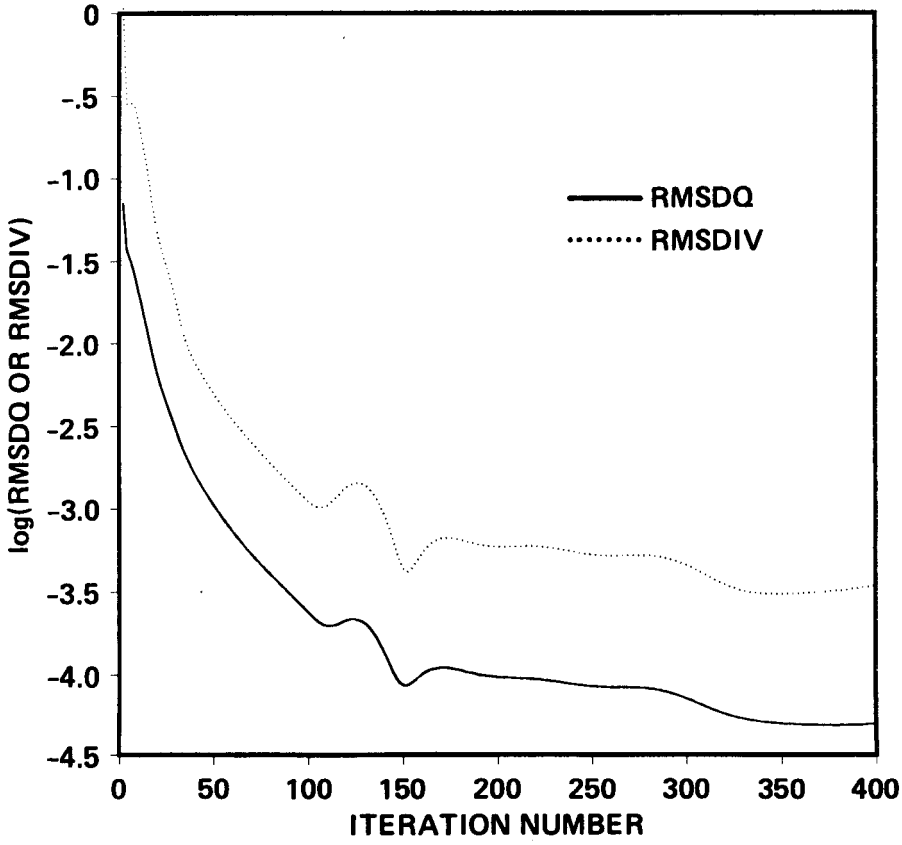


Figure 4.- Convergence history for flow over a circular cylinder.

the two plates. The calculations used one of the flat plates in the xy plane at z = 0 and a symmetry boundary condition in the xy plane at z = 5 post diameters. For this flow calculation, a series of 29 identical 35-by-41 O-grids similar to the one described for the two-dimensional circular cylinder were used parallel to the flat plate, with clustering close to the flat plate. The inflow was prescribed to be a partially-developed boundary layer flow between the flat plates.

The relevant parameters used in this calculation are given in the following list.

BETA	= 5	KPERI	= 1
DTAU	= 0.05	LMAX	= 29
ENDACC	= 1	SMU	= 0.1
IORTHO	= 2	SMUIM	= 0.3
JMAX	= 35	SMUPRS	= 0.1
KMAX	= 41		

In Figure 5, the geometry is shown along with the velocity vectors denoting the inflow velocity profile. The periodic boundary can be seen on the downstream side of the post.

The program output for this example is included here. The first item printed to the output includes the values of all the variables in namelist DATAIN. Next, the convergence history is printed every five iterations. Note that the format of the output was changed here from that in the code in order to fit it on the page. The entry in the first column is NT, which is the current iteration number. The second entry, RMSDQ, is the rms of the change in all four variables at all

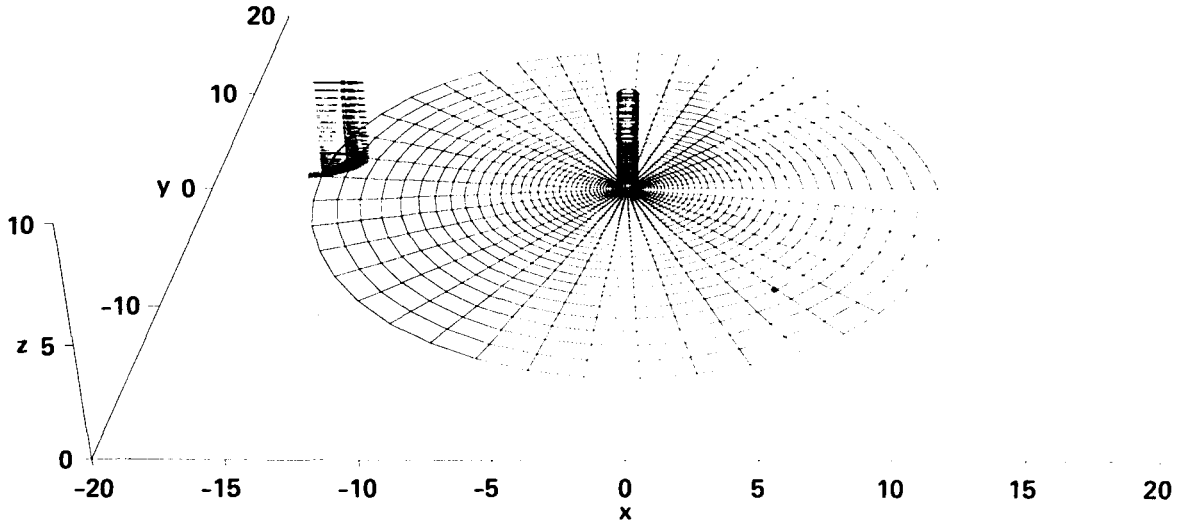


Figure 5.- Three-dimensional post mounted normal to a flat plate with parabolic inlet velocity profile.

the grid points. The entry RMSCO is the rms of the right-hand side of the continuity equation. This includes the divergence of the velocity field plus the smoothing terms. The fourth entry is RMSDIV, which is the rms of the divergence of the velocity field. The entry DQMAX is the maximum change of any of the variables occurring in that iteration, and the J, K, and L values give the location of the maximum change.

Both RMSDQ and DQMAX are measures of convergence. A solution is considered converged if these quantities decrease by three to four orders of magnitude over their original values. In this run, RMSDQ and DQMAX have both converged over three orders in 600 iterations. RMSDIV is a measure of how well the velocity field satisfies the divergence-free condition. This code can usually reduce RMSDIV to between 10^{-2} and 10^{-3}. To achieve this reduction in the present case, it was necessary to reduce the amount of explicit smoothing added to the continuity equation. To do this, one can just change the value of SMUARY(1) in subroutine SMOOTH. This is initially set equal to the input variable SMUPRS. In this calculation, SMUPRS is initially set equal to 0.5. It can be seen that as the calculations progress, RMSCO continues to converge while RMSDIV approaches a constant. This indicates that the smoothing is dominating the continuity equation and should be reduced. The value of SMUARY(1) was decreased by a factor of two at time steps of 350 and 450. The net effect was to decrease the divergence further. It should be noted that if the smoothing is decreased too much, the pressure field will start to develop kinks and the calculation could become unstable.

&DATAIN DISKOUT = .T., PLOT3D = .F., JMAX = 35, KMAX = 41, LMAX = 29, NTMAX = 600, NPRNT = 600, IPRNT = 5, TIMACC = 1, ENDACC = 1, DTAU = 5.E-2, BETA = 5., SMU = 0.1, SMUIM = 0.3, SMUPRS = 0.5, REYNUM = 100., DXDT = 0., RL1 = 1., KPERI = 1, ISTART = 0, IBLKDIA = 2, IMPSMO = 2, &END

NT	RMSDQ	RMSCO	RMSDIV	DQMAX	J	K	L
5	0.2590E-01	0.6427E-01	0.1865E+00	0.3160E+00	2	22	21
10	0.2643E-01	0.7407E-01	0.2856E+00	-0.3742E+00	2	12	19
15	0.2471E-01	0.6486E-01	0.2394E+00	0.3503E+00	2	35	15
20	0.2211E-01	0.5559E-01	0.2212E+00	0.4163E+00	2	11	16
25	0.1747E-01	0.4495E-01	0.1856E+00	0.2632E+00	2	30	18
30	0.1359E-01	0.3389E-01	0.1383E+00	0.1589E+00	2	29	28
35	0.1052E-01	0.2561E-01	0.9981E-01	-0.1227E+00	2	13	16
40	0.7825E-02	0.1923E-01	0.7474E-01	0.8304E-01	6	21	20
45	0.5990E-02	0.1467E-01	0.6023E-01	0.6616E-01	2	31	15
50	0.4737E-02	0.1146E-01	0.5052E-01	0.5518E-01	2	14	17
55	0.3679E-02	0.8812E-02	0.3993E-01	0.4342E-01	10	22	28
60	0.2918E-02	0.6932E-02	0.3002E-01	0.3717E-01	11	22	28
65	0.2343E-02	0.5345E-02	0.2278E-01	0.3016E-01	12	22	28
70	0.1897E-02	0.4236E-02	0.1959E-01	0.2410E-01	13	22	28
75	0.1594E-02	0.3451E-02	0.1788E-01	0.1939E-01	14	22	28
80	0.1333E-02	0.2733E-02	0.1578E-01	0.1584E-01	15	22	28
85	0.1121E-02	0.2210E-02	0.1336E-01	0.1310E-01	16	21	28
90	0.9644E-03	0.1783E-02	0.1145E-01	0.1083E-01	17	21	28
95	0.8348E-03	0.1445E-02	0.1075E-01	0.8746E-02	18	21	28
100	0.7383E-03	0.1228E-02	0.1066E-01	-0.6955E-02	7	38	28
105	0.6612E-03	0.1057E-02	0.1036E-01	-0.6311E-02	7	38	28
110	0.5952E-03	0.9153E-03	0.9791E-02	-0.5887E-02	8	38	28
115	0.5403E-03	0.8070E-03	0.9315E-02	-0.5473E-02	8	38	28
120	0.4921E-03	0.7120E-03	0.9122E-02	-0.5041E-02	8	38	28
125	0.4506E-03	0.6385E-03	0.9082E-02	-0.4583E-02	8	38	28
130	0.4153E-03	0.5804E-03	0.8995E-02	-0.4113E-02	8	38	28
135	0.3844E-03	0.5278E-03	0.8830E-02	-0.3678E-02	9	38	28
140	0.3570E-03	0.4825E-03	0.8675E-02	-0.3334E-02	9	38	28
145	0.3326E-03	0.4416E-03	0.8581E-02	-0.3005E-02	9	38	28
150	0.3114E-03	0.4064E-03	0.8525E-02	-0.2773E-02	16	1	28
155	0.2933E-03	0.3792E-03	0.8466E-02	-0.2560E-02	16	1	28
160	0.2784E-03	0.3594E-03	0.8394E-02	-0.2364E-02	17	1	28
165	0.2659E-03	0.3459E-03	0.8329E-02	-0.2197E-02	17	1	28
170	0.2554E-03	0.3369E-03	0.8280E-02	-0.2032E-02	17	1	28
175	0.2466E-03	0.3321E-03	0.8241E-02	-0.1868E-02	17	1	28
180	0.2394E-03	0.3318E-03	0.8202E-02	-0.1722E-02	18	1	28
185	0.2339E-03	0.3357E-03	0.8162E-02	-0.1584E-02	18	1	28
190	0.2302E-03	0.3435E-03	0.8125E-02	-0.1449E-02	18	1	28
195	0.2283E-03	0.3546E-03	0.8094E-02	-0.1330E-02	19	1	28
200	0.2282E-03	0.3686E-03	0.8068E-02	-0.1258E-02	22	1	28
205	0.2295E-03	0.3848E-03	0.8044E-02	-0.1246E-02	22	1	28
210	0.2319E-03	0.4019E-03	0.8024E-02	-0.1201E-02	21	1	28
215	0.2345E-03	0.4176E-03	0.8008E-02	-0.1130E-02	21	1	28
220	0.2363E-03	0.4301E-03	0.7997E-02	-0.1035E-02	21	1	28
225	0.2366E-03	0.4375E-03	0.7990E-02	-0.9277E-03	20	1	28
230 .	0.2349E-03	0.4392E-03	0.7984E-02	-0.8438E-03	2	8	27
235	0.2312E-03	0.4352E-03	0.7982E-02	-0.8612E-03	2	8	27

NT	RMSDQ	RMSCO	RMSDIV	DQMAX	J	K	L
240	0.2259E-03	0.4266E-03	0.7983E-02	-0.8738E-03	2	8	27
245	0.2194E-03	0.4148E-03	0.7989E-02	-0.8813E-03	2	8	28
250	0.2124E-03	0.4014E-03	0.7999E-02	-0.8814E-03	2	8	28
255	0.2054E-03	0.3875E-03	0.8014E-02	-0.8692E-03	2	8	28
260	0.1987E-03	0.3742E-03	0.8033E-02	-0.8422E-03	2	8	28
265	0.1926E-03	0.3621E-03	0.8056E-02	-0.8018E-03	3	9	28
270	0.1871E-03	0.3513E-03	0.8081E-02	-0.7584E-03	3	34	28
275	0.1821E-03	0.3416E-03	0.8107E-02	-0.7067E-03	3	34	28
280	0.1774E-03	0.3324E-03	0.8133E-02	-0.6500E-03	3	34	28
285	0.1730E-03	0.3235E-03	0.8157E-02	-0.6020E-03	6	39	28
290	0.1685E-03	0.3143E-03	0.8180E-02	-0.5649E-03	6	39	28
295	0.1639E-03	0.3047E-03	0.8200E-02	-0.5304E-03	7	39	28
300	0.1590E-03	0.2943E-03	0.8218E-02	-0.5028E-03	7	39	28
305	0.1539E-03	0.2833E-03	0.8233E-02	-0.4749E-03	7	39	28
310	0.1484E-03	0.2717E-03	0.8246E-02	-0.4461E-03	7	39	28
315	0.1428E-03	0.2595E-03	0.8257E-02	-0.4187E-03	8	39	28
320	0.1371E-03	0.2470E-03	0.8265E-02	-0.4151E-03	34	1	28
325	0.1314E-03	0.2345E-03	0.8273E-02	-0.4132E-03	34	1	28
330	0.1259E-03	0.2223E-03	0.8279E-02	-0.4127E-03	34	1	28
335	0.1206E-03	0.2105E-03	0.8285E-02	-0.4134E-03	34	1	28
340	0.1156E-03	0.1993E-03	0.8291E-02	-0.4148E-03	34	1	28
345	0.1108E-03	0.1888E-03	0.8296E-02	-0.4160E-03	34	1	28
350	0.2740E-03	0.1046E-02	0.8301E-02	0.5572E-02	2	27	24
355	0.1444E-03	0.3286E-03	0.4654E-02	-0.1646E-02	3	13	19
360	0.1197E-03	0.2430E-03	0.4369E-02	0.9543E-03	34	21	4
365	0.1081E-03	0.2149E-03	0.4534E-02	-0.1044E-02	2	21	15
370	0.1031E-03	0.1923E-03	0.4494E-02	-0.6718E-03	2	21	28
375	0.9620E-04	0.1697E-03	0.4496E-02	-0.5160E-03	7	21	19
380	0.8872E-04	0.1475E-03	0.4491E-02	-0.4101E-03	9	21	20
385	0.8326E-04	0.1302E-03	0.4464E-02	-0.3655E-03	34	1	28
390	0.7902E-04	0.1187E-03	0.4466E-02	-0.3348E-03	34	1	28
395	0.7593E-04	0.1091E-03	0.4488E-02	-0.3161E-03	34	1	2
400	0.7371E-04	0.1037E-03	0.4496E-02	-0.3094E-03	34	1	3
405	0.7217E-04	0.1011E-03	0.4487E-02	0.2999E-03	34	41	23
410	0.7105E-04	0.9960E-04	0.4480E-02	0.3009E-03	34	41	23
415	0.7020E-04	0.9798E-04	0.4485E-02	0.3018E-03	34	1	23
420	0.6984E-04	0.9651E-04	0.4496E-02	0.3027E-03	34	1	23
425	0.7039E-04	0.9653E-04	0.4507E-02	0.3030E-03	34	1	23
430	0.7201E-04	0.9873E-04	0.4516E-02	0.3025E-03	34	1	23
435	0.7402E-04	0.1018E-03	0.4524E-02	0.3230E-03	2	22	19
440	0.7520E-04	0.1029E-03	0.4532E-02	0.3490E-03	2	21	28
445	0.7460E-04	0.1001E-03	0.4536E-02	0.3544E-03	4	34	17
450	0.1574E-03	0.5739E-03	0.4537E-02	0.3060E-02	3	27	23
455	0.9444E-04	0.2107E-03	0.2630E-02	0.1044E-02	2	22	23
460	0.7930E-04	0.1491E-03	0.2430E-02	0.6952E-03	4	21	28
465	0.6833E-04	0.1174E-03	0.2499E-02	-0.5334E-03	2	21	16
470	0.6339E-04	0.1012E-03	0.2459E-02	-0.3949E-03	2	21	27
475	0.6121E-04	0.9602E-04	0.2449E-02	-0.3006E-03	7	21	20
480	0.5949E-04	0.9406E-04	0.2451E-02	0.2932E-03	34	1	24
485	0.5911E-04	0.9624E-04	0.2441E-02	0.2958E-03	34	1	24
490	0.5994E-04	0.1017E-03	0.2441E-02	0.2991E-03	34	1	24

NT	RMSDQ	RMSCO	RMSDIV	DQMAX	J	K	L
495	0.6120E-04	0.1064E-03	0.2452E-02	0.3032E-03	34	1	25
500	0.6211E-04	0.1105E-03	0.2457E-02	0.3084E-03	33	1	26
505	0.6255E-04	0.1135E-03	0.2453E-02	0.3124E-03	33	1	26
510	0.6240E-04	0.1150E-03	0.2448E-02	0.3146E-03	34	1	26
515	0.6119E-04	0.1136E-03	0.2448E-02	0.3161E-03	34	1	26
520	0.5852E-04	0.1080E-03	0.2449E-02	0.3156E-03	34	1	28
525	0.5452E-04	0.9839E-04	0.2449E-02	0.3124E-03	34	1	28
530	0.4991E-04	0.8596E-04	0.2448E-02	0.3062E-03	34	1	28
535	0.4572E-04	0.7300E-04	0.2449E-02	0.2968E-03	34	1	28
540	0.4268E-04	0.6205E-04	0.2452E-02	0.2848E-03	34	1	28
545	0.4083E-04	0.5468E-04	0.2455E-02	0.2710E-03	34	1	28
550	0.3970E-04	0.5076E-04	0.2458E-02	0.2563E-03	34	1	28
555	0.3876E-04	0.4909E-04	0.2459E-02	0.2417E-03	34	1	28
560	0.3782E-04	0.4871E-04	0.2460E-02	0.2282E-03	34	1	28
565	0.3696E-04	0.4933E-04	0.2460E-02	0.2162E-03	34	1	28
570	0.3630E-04	0.5075E-04	0.2461E-02	0.2064E-03	34	1	28
575	0.3585E-04	0.5254E-04	0.2462E-02	0.1987E-03	34	1	28
580	0.3550E-04	0.5415E-04	0.2463E-02	0.1933E-03	34	1	28
585	0.3512E-04	0.5515E-04	0.2464E-02	0.1899E-03	34	1	28
590	0.3461E-04	0.5531E-04	0.2464E-02	0.1884E-03	34	1	28
595	0.3390E-04	0.5461E-04	0.2465E-02	0.1885E-03	34	1	28
600	0.3299E-04	0.5311E-04	0.2465E-02	0.1898E-03	34	1	28

REFERENCES

1. Kwak, D.; Chang, J. L. C.; Shanks, S. P.; and Chakravarthy, S.: A Three-Dimensional Incompressible Navier-Stokes Flow Solver Using Primitive Variables. AIAA J., vol 24, no. 3, 390-396, Mar. 1986

2. Chorin, A. J.: A Numerical Method for Solving Incompressible Viscous Flow Problems. J. Comp. Phys., vol. 2, 1967, pp. 12-26.

3. Steger, J. L.; and Kutler, P.: Implicit Finite-Difference Procedures for the Computation of Vortex Wakes. AIAA J., vol. 15, no. 4, 1977, pp. 581-590.

4. Chang, J. L. C.; and Kwak, D.: On the Method of Pseudo Compressibility for Numerically Solving Incompressible Flows. AIAA Paper 84-0252, 1984.

5. Beam, R. M.; and Warming, R. F.: An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation-Law Form. J. Comp. Phys., vol. 22, 1976, pp. 87–110.

6. Briley, W. R.; and McDonald, H.: Solution of the Three-dimensional Compressible Navier-Stokes Equations by an Implicit Technique. Proceedings of the Fourth International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, vol. 35, Springer-Verlag, New York 1975, pp. 105-110.

7. Rogers, S. E.; Chang, J. L. C.; and Kwak, D: A Diagonal Algorithm for the Method of Pseudocompressibility. AIAA Paper 86-1060, 1986.

8. Rogers, S. E.; Kwak, D.; and Kaul, U.: On the Accuracy of the Pseudocompressibility Method in Solving the Incompressible Navier-Stokes Equations. Appl. Math. Modelling, vol. 11, 1987, pp. 35-44.

9. Zilliac, G.: A Computational/Experimental Study of the Flow Around a Body of Revolution at Angle of Attack. NASA TM-88329, 1986.

10. Champney, J. M.; Coakley, T. J.; and Kaul, U. K.: Incompressible Viscous Flow Simulations of the NFAC Wind Tunnel. ATM-FR-25-012, 1986.

11. Kaul, U. K.; Kwak, D.; and Wagner, C: A Computational Study of Saddle Point Separation and Horseshoe Vortex System. AIAA Paper 85-0182, 1985.

12. Rogers, S. E.; Kwak, D.; and Kaul, U. K.: A Numerical Study of Three-Dimensionsal Incompressible Flow Around Multiple Posts. AIAA Paper 86-0353, 1986.

13. Chang, J. L. C.; Kwak, D.; and Dao, S. C.: A Three Dimensional Incompressible Flow Simulation Method and its Application to the Space Shuttle Main Engine, Part 1, Laminar Flow. AIAA Paper 85-0175, 1985.

14. Yang, R. J.; Chang, J. L. C.; and Kwak, D.: A Navier-Stokes Simulation of the Space Shuttle Main Engin Hot Gas Manifold. AIAA Paper 87-0368, 1987.

15. Armaly, B. F.; Durst, F.; Pereira, J. C. F.; and Schönung, B.: Experimental and Theoretical Investigation of Backward Facing Step Flow. J. Fluid Mech., vol. 127, 1983, pp. 473-496.

16. Warming, R. F.; Beam, R. M.; and Hyett, B. J.: Diagonalization and Simultaneous Symmetrization of the Gas-Dynamic Matrices. Mathematics of Computation, vol. 29, 1975, pp. 1037-1045.

17. Turkel, E.: Symmetrization of the Fluid Dynamic Matrices with Applications, Mathematics of Computation. vol. 27, 1973, pp. 729-736.

18. Pulliam, T. H.; and Chaussee, D. S.: A Diagonal Form of an Implicit Approximate-Factorization Algorithm. J. Comp. Phys., vol. 39, 1981, pp. 347-363.

19. Flores, J.: Convergence Acceleration for a Three-Dimensional Euler/Navier-Stokes Zonal Approach. AIAA Paper 85-1495, 1985.

APPENDIX
THEORETICAL FOUNDATION

I. Governing Equations

Incompressible Navier-Stokes Equations

Unsteady, three-dimensional, viscous, incompressible flow with constant density is governed by the following Navier-Stokes equations:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{2}$$

where t is the time, x_i the Cartesian coordinates, u_i the corresponding velocity components, p the pressure, and τ_{ij} the viscous stress tensor. All of the variables are nondimensionalized by a reference velocity and length scale. The viscous stress tensor can be written as

$$\tau_{ij} = 2\nu S_{ij} - R_{ij} \tag{3}$$

Here, ν is the kinematic viscosity, S_{ij} is the strain rate tensor, and R_{ij} is the Reynolds stresses. The strain rate tensor is defined by

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{4}$$

Various levels of closure models for R_{ij} are possible. In the present code, turbulence is simulated by an eddy viscosity model, using a constitutive equation of the following form:

$$R_{ij} = \frac{1}{3}R_{kk}\delta_{ij} - 2\nu_t S_{ij} \tag{5}$$

where ν_t is the turbulent eddy viscosity. By including the normal stress, R_{kk}, in the pressure, the variable ν in equation (3) can be replaced by $(\nu + \nu_t)$.

$$\tau_{ij} = 2(\nu + \nu_t)S_{ij} = 2\nu_T S_{ij} \tag{6}$$

In the remainder of this paper, the total viscosity, ν_T, will be represented simply by ν. Therefore, the incompressible Navier-Stokes equations are modified to allow variable viscosity in the present formulation.

Pseudocompressible Formulation

The system of equations given in the preceding section is not as readily solvable by a fast alternating-direction-implicit (ADI) scheme as the unsteady compressible Navier-Stokes equations

which are hyperbolic. This difficulty can be alleviated by modifying the continuity equation as follows:

$$\frac{1}{\beta}\frac{\partial p}{\partial t} + \frac{\partial u_i}{\partial x_i} = 0 \tag{7}$$

By combining equations (2) and (7), the governing equations are written as one vector equation

$$\frac{\partial}{\partial t}D + \frac{\partial}{\partial x}(E - E_v) + \frac{\partial}{\partial y}(F - F_v) + \frac{\partial}{\partial z}(G - G_v) = 0 \tag{8}$$

where

$$D = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix}$$

$$E = \begin{bmatrix} \beta u \\ u^2 + p \\ uv \\ uw \end{bmatrix} \quad F = \begin{bmatrix} \beta v \\ vu \\ v^2 + p \\ vw \end{bmatrix} \quad G = \begin{bmatrix} \beta w \\ wu \\ wv \\ w^2 + p \end{bmatrix} \tag{9}$$

$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \end{bmatrix} \quad F_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \end{bmatrix} \quad G_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \end{bmatrix}.$$

Coordinate Transformation

To perform calculations on three-dimensional arbitrarily shaped geometries, the following generalized independent variables are introduced which transform the physical coordinates into general curvilinear coordinates.

$$\begin{aligned} \tau &= t \\ \xi &= \xi(x, y, z, t) \\ \eta &= \eta(x, y, z, t) \\ \zeta &= \zeta(x, y, z, t) \end{aligned} \tag{10}$$

The spatial and time derivatives become

$$\begin{aligned} \frac{\partial}{\partial x_i} = \frac{\partial \xi}{\partial x_i}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x_i}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial x_i}\frac{\partial}{\partial \zeta} = \frac{\partial \xi_j}{\partial x_i}\frac{\partial}{\partial \xi_j} \\ \frac{\partial}{\partial t} = \frac{\partial}{\partial \tau} + \frac{\partial \xi_j}{\partial t}\frac{\partial}{\partial \xi_j} \end{aligned} \tag{11}$$

Here the Jacobian of the transformation is defined as

$$J = \det\frac{\partial(\xi,\eta,\zeta)}{\partial(x,y,z)} = \begin{vmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{vmatrix} \tag{12}$$

where, for example,

$$\xi_x = \frac{\partial\xi}{\partial x},\ \eta_y = \frac{\partial\eta}{\partial y}$$

In actual coding, metric terms are calculated as follows :

$$\begin{pmatrix} \xi_x \\ \xi_y \\ \xi_z \end{pmatrix} = \frac{1}{J'}\begin{pmatrix} y_\eta z_\zeta - y_\zeta z_\eta \\ x_\zeta z_\eta - x_\eta z_\zeta \\ x_\eta y_\zeta - x_\zeta y_\eta \end{pmatrix}$$

$$\begin{pmatrix} \eta_x \\ \eta_y \\ \eta_z \end{pmatrix} = \frac{1}{J'}\begin{pmatrix} y_\zeta z_\xi - y_\xi z_\zeta \\ x_\xi z_\zeta - x_\zeta z_\xi \\ x_\zeta y_\xi - x_\xi y_\zeta \end{pmatrix} \tag{13}$$

where

$$\begin{pmatrix} \zeta_x \\ \zeta_y \\ \zeta_z \end{pmatrix} = \frac{1}{J'}\begin{pmatrix} y_\xi z_\eta - y_\eta z_\xi \\ x_\eta z_\xi - x_\xi z_\eta \\ x_\xi y_\eta - x_\eta y_\xi \end{pmatrix}$$

$$J' = \det\frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)} = \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix} \tag{14}$$

Applying the transformation to equation (8) yields

$$\frac{\partial}{\partial\tau}\hat{D} + \frac{\partial}{\partial\xi}(\hat{E} - \hat{E}_v) + \frac{\partial}{\partial\eta}(\hat{F} - \hat{F}_v) + \frac{\partial}{\partial\zeta}(\hat{G} - \hat{G}_v) = 0 \tag{15}$$

where

$$\hat{D} = \frac{D}{J}$$

$$\hat{E} = \frac{1}{J}\left[\xi_t D + \xi_x E + \xi_y F + \xi_z G\right]$$

$$\hat{F} = \frac{1}{J}\left[\eta_t D + \eta_x E + \eta_y F + \eta_z G\right]$$

$$\hat{G} = \frac{1}{J}\left[\zeta_t D + \zeta_x E + \zeta_y F + \zeta_z G\right]$$

$$\hat{E}_v = \frac{1}{J}\left[\quad \xi_x E_v + \xi_y F_v + \xi_z G_v\right]$$

$$\hat{F}_v = \frac{1}{J}\left[\quad \eta_x E_v + \eta_y F_v + \eta_z G_v\right]$$

$$\hat{G}_v = \frac{1}{J}\left[\quad \zeta_x E_v + \zeta_y F_v + \zeta_z G_v\right]$$

The contravariant velocities, U, V, and W without metric normalization are defined as

$$\begin{aligned} U &= \xi_t + \xi_x u + \xi_y v + \xi_z w \\ V &= \eta_t + \eta_x u + \eta_y v + \eta_z w \\ W &= \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w \end{aligned} \tag{16}$$

The flux vectors can be rewritten as

$$\begin{aligned} \hat{E} &= \frac{1}{J}\begin{bmatrix} \beta U + \xi_t(p-\beta) \\ uU + \xi_x p \\ vU + \xi_y p \\ wU + \xi_z p \end{bmatrix} \\ \hat{F} &= \frac{1}{J}\begin{bmatrix} \beta V + \eta_t(p-\beta) \\ uV + \eta_x p \\ vV + \eta_y p \\ wV + \eta_z p \end{bmatrix} \\ \hat{G} &= \frac{1}{J}\begin{bmatrix} \beta W + \zeta_t(p-\beta) \\ uW + \zeta_x p \\ vW + \zeta_y p \\ wW + \zeta_z p \end{bmatrix} \end{aligned} \tag{17}$$

Viscous Terms

The viscous terms are quite lengthy and therefore are given here separately.

$$\begin{aligned} \frac{\partial}{\partial x_j}\tau_{ij} &= \frac{\partial}{\partial x_j}\nu S_{ij} \\ &= \frac{\partial}{\partial x}\begin{bmatrix} u_x + u_x \\ v_x + u_y \\ w_x + u_z \end{bmatrix} + \frac{\partial}{\partial y}\begin{bmatrix} u_y + v_x \\ v_y + v_y \\ w_y + v_z \end{bmatrix} + \frac{\partial}{\partial z}\begin{bmatrix} u_z + w_x \\ v_z + w_y \\ w_z + w_z \end{bmatrix} \end{aligned}$$

where, for example,

$$u_x = \frac{\partial u}{\partial x},\ u_y = \frac{\partial u}{\partial y}$$

When ν is constant, the contribution of the second terms in each bracket sums up to zero when the velocity field is divergence-free. However, since in general, ν varies in space and time these terms have to be kept. Then the viscous terms in transformed coordinates are given by

$$\hat{E}_v = \frac{\nu}{J}\left\{ \nabla\xi \cdot \left(\nabla\xi\frac{\partial}{\partial\xi} + \nabla\eta\frac{\partial}{\partial\eta} + \nabla\zeta\frac{\partial}{\partial\zeta}\right)\begin{bmatrix} 0 \\ u \\ v \\ w \end{bmatrix} + \left(\xi_x\frac{\partial u}{\partial\xi_i} + \xi_y\frac{\partial v}{\partial\xi_i} + \xi_z\frac{\partial w}{\partial\xi_i}\right)\begin{bmatrix} 0 \\ \frac{\partial}{\partial x}\xi_i \\ \frac{\partial}{\partial y}\xi_i \\ \frac{\partial}{\partial z}\xi_i \end{bmatrix} \right\}$$

$$\hat{F}_v = \frac{\nu}{J}\left\{(\nabla\eta\cdot(\nabla\xi\frac{\partial}{\partial\xi}+\nabla\eta\frac{\partial}{\partial\eta}+\nabla\zeta\frac{\partial}{\partial\zeta})\begin{bmatrix}0\\u\\v\\w\end{bmatrix}+(\eta_x\frac{\partial u}{\partial\xi_i}+\eta_y\frac{\partial v}{\partial\xi_i}+\eta_z\frac{\partial w}{\partial\xi_i})\begin{bmatrix}0\\\frac{\partial}{\partial x}\xi_i\\\frac{\partial}{\partial y}\xi_i\\\frac{\partial}{\partial z}\xi_i\end{bmatrix}\right\}$$

$$\hat{G}_v = \frac{\nu}{J}\left\{\nabla\zeta\cdot(\nabla\xi\frac{\partial}{\partial\xi}+\nabla\eta\frac{\partial}{\partial\eta}+\nabla\zeta\frac{\partial}{\partial\zeta})\begin{bmatrix}0\\u\\v\\w\end{bmatrix}+(\zeta_x\frac{\partial u}{\partial\xi_i}+\zeta_y\frac{\partial v}{\partial\xi_i}+\zeta_z\frac{\partial w}{\partial\xi_i})\begin{bmatrix}0\\\frac{\partial}{\partial x}\xi_i\\\frac{\partial}{\partial y}\xi_i\\\frac{\partial}{\partial z}\xi_i\end{bmatrix}\right\} \quad (18a)$$

The viscous terms given in equation (18a) can be simplified under special conditions. If an orthogonal coordinate system is used, then

$$\nabla\xi_i\cdot\nabla\xi_j = 0 \qquad \text{for } i\neq j$$

Equation (18a) becomes

$$\hat{E}_v = \frac{\nu}{J}\left\{(\xi_x{}^2+\xi_y{}^2+\xi_z{}^2)Im\frac{\partial D}{\partial\xi}+(\xi_x\frac{\partial u}{\partial\xi_i}+\xi_y\frac{\partial v}{\partial\xi_i}+\xi_z\frac{\partial w}{\partial\xi_i})\begin{bmatrix}0\\\frac{\partial}{\partial x}\xi_i\\\frac{\partial}{\partial y}\xi_i\\\frac{\partial}{\partial z}\xi_i\end{bmatrix}\right\}$$

$$\hat{F}_v = \frac{\nu}{J}\left\{(\eta_x{}^2+\eta_y{}^2+\eta_z{}^2)Im\frac{\partial D}{\partial\eta}+(\eta_x\frac{\partial u}{\partial\xi_i}+\eta_y\frac{\partial v}{\partial\xi_i}+\eta_z\frac{\partial w}{\partial\xi_i})\begin{bmatrix}0\\\frac{\partial}{\partial x}\xi_i\\\frac{\partial}{\partial y}\xi_i\\\frac{\partial}{\partial z}\xi_i\end{bmatrix}\right\} \quad (18b)$$

$$\hat{G}_v = \frac{\nu}{J}\left\{(\zeta_x{}^2+\zeta_y{}^2+\zeta_z{}^2)Im\frac{\partial D}{\partial\zeta}+(\zeta_x\frac{\partial u}{\partial\xi_i}+\zeta_y\frac{\partial v}{\partial\xi_i}+\zeta_z\frac{\partial w}{\partial\xi_i})\begin{bmatrix}0\\\frac{\partial}{\partial x}\xi_i\\\frac{\partial}{\partial y}\xi_i\\\frac{\partial}{\partial z}\xi_i\end{bmatrix}\right\}$$

where

$$Im = \begin{bmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

For constant ν (i.e., laminar flow), the contribution by the second term in parentheses in equation (18b) will approach zero as the steady state is reached, where equation (1) is numerically satisfied. Therefore, for flow with constant ν in orthogonal coordinates, equation (18b) can be reduced further as follows:

$$\hat{E}_v = \left(\frac{\nu}{J}\right)\left(\xi_x{}^2 + \xi_y{}^2 + \xi_z{}^2\right) Im \frac{\partial D}{\partial \xi} = \gamma_1 D$$
$$\hat{F}_v = \left(\frac{\nu}{J}\right)\left(\eta_x{}^2 + \eta_y{}^2 + \eta_z{}^2\right) Im \frac{\partial D}{\partial \eta} = \gamma_2 D \tag{18c}$$
$$\hat{G}_v = \left(\frac{\nu}{J}\right)\left(\zeta_x{}^2 + \zeta_y{}^2 + \zeta_z{}^2\right) Im \frac{\partial D}{\partial \zeta} = \gamma_3 D$$

II. Numerical Algorithm

Time-Advancing Algorithm

The numerical algorithm used to advance equation (15) in time is an implicit, noniterative, approximately factored, finite-difference scheme described by both Beam and Warming (ref. 5) and Briley and McDonald (ref. 6). The time-differencing used by this scheme, generally known as the trapezoidal rule, is given by

$$\hat{D}^{n+1} = \hat{D}^n + \frac{\Delta\tau}{2}\left[\left(\frac{\partial \hat{D}}{\partial \tau}\right)^n + \left(\frac{\partial \hat{D}}{\partial \tau}\right)^{n+1}\right] + O(\Delta\tau^3) \tag{19}$$

where the superscript n refers to the nth time-step. The finite-difference form of equation (15) is

$$\delta_\tau \hat{D} + \delta_\xi(\hat{E} - \hat{E}_v) + \delta_\eta(\hat{F} - \hat{F}_v) + \delta_\zeta(\hat{G} - \hat{G}_v) = 0 \tag{20}$$

where, for example, δ_ξ is the finite difference operator for $\frac{\partial}{\partial \xi}$. Expressions for $\hat{E}$, $\hat{F}$, and $\hat{G}$ are given by equation (17); expresions for $\hat{E}_v$, $\hat{F}_v$, and $\hat{G}_v$ are given by equation (18a) for a nonorthogonal grid, by equation (18b) when an orthogonal grid is assumed, or by equation (18c) for the case of laminar flow and an orthogonal grid. By substituting equation (20) into equation (19) and using $D = \hat{D}J$, one obtains

$$\begin{aligned} D^{n+1} + \frac{\Delta\tau}{2} J^{n+1}\left[\delta_\xi(\hat{E} - \hat{E}_v)^{n+1} + \delta_\eta(\hat{F} - \hat{F}_v)^{n+1} + \delta_\zeta(\hat{G} - \hat{G}_v)^{n+1}\right] \\ = \frac{J^{n+1}D^n}{J^n} - \frac{\Delta\tau}{2} J^{n+1}\left[\delta_\xi(\hat{E} - \hat{E}_v)^n + \delta_\eta(\hat{F} - \hat{F}_v)^n + \delta_\zeta(\hat{G} - \hat{G}_v)^n\right] \end{aligned} \tag{21}$$

Solving for D^{n+1} is nonlinear in nature since $\hat{E}^{n+1} = \hat{E}(D^{n+1})$ is a nonlinear function of D^{n+1} as are $\hat{F}^{n+1}$ and $\hat{G}^{n+1}$. The following linearization procedure is used. A local Taylor expansion about u^n yields

$$\begin{aligned} \hat{E}^{n+1} &= \hat{E}^n + \hat{A}^n(D^{n+1} - D^n) + O(\Delta\tau^2) \\ \hat{F}^{n+1} &= \hat{F}^n + \hat{B}^n(D^{n+1} - D^n) + O(\Delta\tau^2) \\ \hat{G}^{n+1} &= \hat{G}^n + \hat{C}^n(D^{n+1} - D^n) + O(\Delta\tau^2) \end{aligned} \tag{22}$$

where $\hat{A}$, $\hat{B}$, and $\hat{C}$ are the Jacobian matrices

$$\hat{A} = \frac{\partial \hat{E}}{\partial D}, \ \hat{B} = \frac{\partial \hat{F}}{\partial D}, \ \hat{C} = \frac{\partial \hat{G}}{\partial D} \tag{23}$$

The Jacobian matrices can all be represented by the following, where $\hat{A}_i = \hat{A}, \hat{B}$, or $\hat{C}$ for $i = 1, 2$, or 3, respectively.

$$\hat{A}_i = \frac{1}{J} \begin{bmatrix} L_0 & L_1\beta & L_2\beta & L_3\beta \\ L_1 & Q + L_1 u & L_2 u & L_3 u \\ L_2 & L_1 v & Q + L_2 v & L_3 v \\ L_3 & L_1 w & L_2 w & Q + L_3 w \end{bmatrix} \tag{24}$$

where

$$Q = L_0 + L_1 u + L_2 v + L_3 w$$
$$L_0 = (\xi_i)_t, \ L_1 = (\xi_i)_x, \ L_2 = (\xi_i)_y, \ L_3 = (\xi_i)_z$$
$$\xi_i = \xi, \ \eta, \text{ or } \zeta \text{ for } \hat{A}_i = \hat{A}, \ \hat{B}, \text{ or } \hat{C}, \text{ respectively}$$

Substituting equation (22) into equation (21) results in the governing equation in delta form

$$\begin{aligned} &\left\{ I + \frac{h}{2} J^{n+1} \left[\delta_\xi (\hat{A}^n - \Gamma_1) + \delta_\eta (\hat{B}^n - \Gamma_2) + \delta_\zeta (\hat{C}^n - \Gamma_3) \right] \right\} (D^{n+1} - D^n) \\ &= -\Delta\tau J^{n+1} \left[\delta_\xi (\hat{E} - \hat{E}_v)^n + \delta_\eta (\hat{F} - \hat{F}_v)^n + \delta_\zeta (\hat{G} - \hat{G}_v)^n \right] \\ &+ \left(\frac{J^{n+1}}{J^n} - 1 \right) D^n \end{aligned} \tag{25}$$

where

$$\Gamma_1 D^{n+1} = \hat{E}_v^{n+1}, \ \Gamma_2 D^{n+1} = \hat{F}_v^{n+1}, \ \Gamma_3 D^{n+1} = \hat{G}_v^{n+1}$$
$$h = \Delta\tau \quad \text{for trapezoidal scheme}$$
$$= 2\Delta\tau \quad \text{for Euler scheme}$$

At this point it should be noted that the notation of the form $[\delta_\xi (A - \Gamma)]D$ refers to $\frac{\partial}{\partial \xi}(AD) - \frac{\partial}{\partial \xi}(\Gamma D)$ and not $\frac{\partial A}{\partial \xi} D - \frac{\partial \Gamma}{\partial \xi} D$.

Approximate Factorization

The solution of equation (25) would involve a formidable matrix-inversion problem. With the use of an ADI-type scheme, the problem could be reduced to the inversion of three matrices of small bandwidth, for which there exists efficient solution algorithms. The particular ADI form used here is known as approximate factorization (AF) (refs. 5,6). It is difficult to implement the AF scheme to solve equation (25) in its full matrix form. Noting that at steady-state, the left-hand side of equation (25) approaches zero, a simplified expression for the viscous term as

shown in equation (18c) is used on the left-hand side. To maintain the accuracy of the solution, the entire viscous term is used on the right-hand side. Using this, the governing equation becomes

$$L_\xi L_\eta L_\zeta (D^{n+1} - D^n) = RHS \tag{26}$$

where

$$\begin{aligned} L_\xi &= \left[I + \frac{\Delta\tau}{2} J^{n+1} \delta_\xi (\hat{A}^n - \gamma_1) \right] \\ L_\eta &= \left[I + \frac{\Delta\tau}{2} J^{n+1} \delta_\eta (\hat{B}^n - \gamma_2) \right] \\ L_\zeta &= \left[I + \frac{\Delta\tau}{2} J^{n+1} \delta_\zeta (\hat{C}^n - \gamma_3) \right] \end{aligned} \tag{27}$$

and RHS is the right-hand side of equation (25). When second-order central differencing is used, the solution to this problem becomes the inversion of three block-tridiagonal matrices. The inversion problem is reduced to the three inversions

$$\begin{aligned} (L_\eta)\Delta\bar{D} &= RHS \\ (L_\xi)\Delta\bar{\bar{D}} &= \Delta\bar{D} \\ (L_\zeta)\Delta D^{n+1} &= \Delta\bar{\bar{D}}. \end{aligned} \tag{28}$$

These inversions are carried out for all interior points, and the boundary conditions can be implemented explicitly. However, it is possible to implement the boundary conditions implicitly, which will be discussed in a later section.

The factorization has introduced the following second-order cross-product term into the equation.

$$h^2 \left[\delta_\xi A \delta_\eta B + \delta_\eta B \delta_\zeta C + \delta_\zeta C \delta_\xi A \right] \Delta D + O(h^3) \tag{29}$$

where

$$A = \hat{A}^n - \gamma_1, \quad B = \hat{B}^n - \gamma_2, \quad C = \hat{C}^n - \gamma_3, \quad h = \frac{\Delta\tau}{2} J^{n+1}$$

To maintain the second-order accuracy of the scheme, the added terms must be kept smaller than the original terms in the equations everywhere in the computational domain. This puts a restriction on the size of the artificial compressibility parameter β. The proper choice of β will be discussed in a later section. When the approximate factorization scheme is applied, the stability of the scheme is dependent on the use of higher-order smoothing terms. These are used to damp out higher frequency oscillations which arise in the solution because of the use of second-order central differencing, and will be discussed in a following section.

Numerical Dissipation / Smoothing

Higher-order smoothing terms are required to make the present algorithm stable. These terms help to damp out the higher order oscillations in the solution that are caused by the use of

second-order central differencing. The smoothing term is derived in conjunction with an upwind finite-difference approximation. The fourth-order smoothing operator is given by

$$(\nabla_x \Delta_x)^2 u = \frac{1}{4\Delta x}(u_{j-2} - 4u_{j-1} + 6u_j - 4u_{j+1} + u_{j+2}) \tag{30}$$

The second-order smoothing operator is given by

$$\nabla_x \Delta_x u = \frac{1}{2\Delta x}(u_{j+1} - 2u_j + u_{j-1}) \tag{31}$$

By including these smoothing terms, equations (26) and (27) become

$$L_\xi L_\eta L_\zeta (D^{n+1} - D^n) = \text{RHS} - \epsilon_e[(\nabla_\xi \Delta_\xi)^2 + (\nabla_\eta \Delta_\eta)^2 + (\nabla_\zeta \Delta_\zeta)^2]D^n \tag{26'}$$

where

$$\begin{aligned} L_\xi &= \left[I + \frac{\Delta\tau}{2} J^{n+1}\delta_\xi(\hat{A}^n - \gamma_1) + \epsilon_i \nabla_\xi \Delta_\xi\right] \\ L_\eta &= \left[I + \frac{\Delta\tau}{2} J^{n+1}\delta_\eta(\hat{B}^n - \gamma_2) + \epsilon_i \nabla_\eta \Delta_\eta\right] \\ L_\zeta &= \left[I + \frac{\Delta\tau}{2} J^{n+1}\delta_\zeta(\hat{C}^n - \gamma_3) + \epsilon_i \nabla_\zeta \Delta_\zeta\right] \end{aligned} \tag{27'}$$

Here ϵ_i is the implicit smoothing coefficient and ϵ_e is the explicit smoothing coefficient.

To preserve the tridiagonal nature of the system, only second-order smoothing can be used on the implicit side of the equation, whereas fourth-order smoothing is used on the right-hand side. However, when the diagonal algorithm is used, the bandwidth of the system can be increased to a pentadiagonal, since the scaler inversions are relatively cheap. This makes it possible to use fourth-order smoothing on the implicit side of the equations as well.

When using the second-order implicit, fourth-order explicit scheme, in order to damp out the numerical fluctuations as time advances, the smoothing parameter must satisfy

$$\epsilon_i > 2\epsilon_e \tag{32}$$

In practice, $\epsilon_i = 3\epsilon_e$ is often used, with $\epsilon_e \approx 0.1$. It sometimes is necessary to change the amount of smoothing applied explicitly to the pressure equation; this will be discussed in a later section.

Diagonal Algorithm

In a diagonal algorithm, a similarity transform is used to diagonalize the Jacobian matrices and uncouple the set of equations. The equations can then be solved by solving scalar tridiagonal matrices instead of solving block tridiagonal matrices. A similarity transform which symmetrizes and diagonalizes the matrices of the compressible gas-dynamic equations has been used by Warming et al. (ref. 16) and Turkel (ref. 17). This method was used by Pulliam and

Chaussee (ref. 18) to produce a diagonal algorithm for the Euler equations. This can be applied to the compressible Navier-Stokes equations to obtain a considerable savings in computing time (ref. 19). In this section, similarity transforms for the matrices used in the method of pseudocompressibility are presented. They are used in a diagonal algorithm which results in a substantial reduction in computer time.

Similarity transformations exist which diagonalize the Jacobian matrices.

$$\hat{A}_i = T_i \hat{\Lambda}_i T_i^{-1} \tag{33}$$

where $\hat{\Lambda}_i$ is a diagonal matrix whose elements are the eigenvalues of the Jacobian matrices. It is given by

$$\hat{\Lambda}_i = \frac{1}{J} \begin{bmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & Q+C & 0 \\ 0 & 0 & 0 & Q-C \end{bmatrix} \tag{34}$$

where C is the pseudospeed of sound, which is given by

$$C = \sqrt{(Q - L_0)^2 + \beta(L_1^2 + L_2^2 + L_3^2)}$$

The T_i matrix is composed of the eigenvectors of the Jacobian matrix. For i=1 (ξ-sweep), the first two eigenvectors are given by

$$\vec{X}_1 = \begin{bmatrix} 0 \\ \tilde{x}_\zeta \\ \tilde{y}_\zeta \\ \tilde{z}_\zeta \end{bmatrix}, \quad \vec{X}_2 = \begin{bmatrix} 0 \\ \tilde{x}_\eta \\ \tilde{y}_\eta \\ \tilde{z}_\eta \end{bmatrix} \tag{35}$$

For i=2 (η-sweep) they are

$$\vec{X}_1 = \begin{bmatrix} 0 \\ \tilde{x}_\xi \\ \tilde{y}_\xi \\ \tilde{z}_\xi \end{bmatrix}, \quad \vec{X}_2 = \begin{bmatrix} 0 \\ \tilde{x}_\zeta \\ \tilde{y}_\zeta \\ \tilde{z}_\zeta \end{bmatrix} \tag{36}$$

For i=3 (ζ-sweep) they are

$$\vec{X}_1 = \begin{bmatrix} 0 \\ \tilde{x}_\eta \\ \tilde{y}_\eta \\ \tilde{z}_\eta \end{bmatrix}, \quad \vec{X}_2 = \begin{bmatrix} 0 \\ \tilde{x}_\xi \\ \tilde{y}_\xi \\ \tilde{z}_\xi \end{bmatrix} \tag{37}$$

The terms in these eigenvectors have been normalized to simplify the inverse matrix with, for example,

$$\tilde{x}_\xi = \frac{x_\xi \sqrt{J}}{[L_1^2 + L_2^2 + L_3^2]^{1/4}}$$

$$\tilde{x}_\eta = \frac{x_\eta \sqrt{J}}{[L_1^2 + L_2^2 + L_3^2]^{1/4}}$$

The eigenvectors associated with the last two eigenvalues are given by

$$\vec{X}_3 = \begin{bmatrix} \tilde{C}(\tilde{C}-\tilde{Q}) \\ u+\tilde{L}_1(\tilde{C}-\tilde{Q}) \\ v+\tilde{L}_2(\tilde{C}-\tilde{Q}) \\ w+\tilde{L}_3(\tilde{C}-\tilde{Q}) \end{bmatrix}$$

$$\vec{X}_4 = \begin{bmatrix} \tilde{C}(\tilde{C}+\tilde{Q}) \\ u-\tilde{L}_1(\tilde{C}+\tilde{Q}) \\ v-\tilde{L}_2(\tilde{C}+\tilde{Q}) \\ w-\tilde{L}_3(\tilde{C}+\tilde{Q}) \end{bmatrix} \tag{38}$$

where

$$\tilde{Q} = \tilde{L}_1 u + \tilde{L}_2 v + \tilde{L}_3 w$$

$$\tilde{C} = \sqrt{\tilde{Q}^2 + \beta}$$

$$\tilde{L}_i = \frac{L_i}{\sqrt{L_1^2 + L_2^2 + L_3^2}}$$

The columns of the T_i matrix are formed with the eigenvectors

$$T_i = [\, \vec{X}_1 \quad \vec{X}_2 \quad \vec{X}_3 \quad \vec{X}_4 \,]$$

The elements of the inverse of the eigenvector matrix are given by

$$(T_i^{-1})_{11} = \frac{1}{\tilde{C}^2}[\tilde{x}_{\xi_{i+1}}(\tilde{L}_2 w - \tilde{L}_3 v) + \tilde{y}_{\xi_{i+1}}(\tilde{L}_3 u - \tilde{L}_1 w) + \tilde{z}_{\xi_{i+1}}(\tilde{L}_1 v - \tilde{L}_2 u)]$$

$$(T_i^{-1})_{21} = \frac{1}{\tilde{C}^2}[\tilde{x}_{\xi_{i+2}}(\tilde{L}_3 v - \tilde{L}_2 w) + \tilde{y}_{\xi_{i+2}}(\tilde{L}_1 w - \tilde{L}_3 u) + \tilde{z}_{\xi_{i+2}}(\tilde{L}_2 u - \tilde{L}_1 v)]$$

$$(T_i^{-1})_{31} = \frac{1}{2\tilde{C}^2}$$

$$(T_i^{-1})_{41} = \frac{1}{2\tilde{C}^2}$$

$$(T_i^{-1})_{12} = \frac{1}{\tilde{C}^2}[\tilde{z}_{\xi_{i+1}}(\tilde{L}_2\beta + \tilde{Q}v) - \tilde{y}_{\xi_{i+1}}(\tilde{L}_3\beta + \tilde{Q}w)]$$

$$(T_i^{-1})_{22} = \frac{1}{\tilde{C}^2}[\tilde{y}_{\xi_{i+2}}(\tilde{L}_3\beta + \tilde{Q}w) - \tilde{z}_{\xi_{i+2}}(\tilde{L}_2\beta + \tilde{Q}v)]$$

$$(T_i^{-1})_{32} = \frac{\tilde{L}_1(\tilde{Q}+\tilde{C})}{2\tilde{C}^2}$$

$$(T_i^{-1})_{42} = \frac{\tilde{L}_1(\tilde{Q}-\tilde{C})}{2\tilde{C}^2}$$

$$
\begin{aligned}
(T_i^{-1})_{13} &= \frac{1}{\tilde{C}^2}[\tilde{x}_{\xi_{i+1}}(\tilde{L}_3\beta + \tilde{Q}w) - \tilde{z}_{\xi_{i+1}}(\tilde{L}_1\beta + \tilde{Q}u)] \\
(T_i^{-1})_{23} &= \frac{1}{\tilde{C}^2}[\tilde{z}_{\xi_{i+2}}(\tilde{L}_1\beta + \tilde{Q}u) - \tilde{x}_{\xi_{i+2}}(\tilde{L}_3\beta + \tilde{Q}w)] \\
(T_i^{-1})_{33} &= \frac{\tilde{L}_2(\tilde{Q} + \tilde{C})}{2\tilde{C}^2} \\
(T_i^{-1})_{43} &= \frac{\tilde{L}_2(\tilde{Q} - \tilde{C})}{2\tilde{C}^2} \\
(T_i^{-1})_{14} &= \frac{1}{\tilde{C}^2}[\tilde{y}_{\xi_{i+1}}(\tilde{L}_1\beta + \tilde{Q}u) - \tilde{x}_{\xi_{i+1}}(\tilde{L}_2\beta + \tilde{Q}v)] \\
(T_i^{-1})_{24} &= \frac{1}{\tilde{C}^2}[\tilde{x}_{\xi_{i+2}}(\tilde{L}_2\beta + \tilde{Q}v) - \tilde{y}_{\xi_{i+2}}(\tilde{L}_1\beta + \tilde{Q}u)] \\
(T_i^{-1})_{34} &= \frac{\tilde{L}_3(\tilde{Q} + \tilde{C})}{2\tilde{C}^2} \\
(T_i^{-1})_{44} &= \frac{\tilde{L}_3(\tilde{Q} - \tilde{C})}{2\tilde{C}^2}
\end{aligned}
\tag{39}
$$

The determinant of T_i is given by

$$det(T_i) = 2\tilde{C}^3$$

which remains bounded independent of the geometry. For more detail on the derivation of these matrices, see ref. 7.

The implementation of the diagonal scheme involves replacing the Jacobian matrices in the implicit operators with the product of the similarity transform matrices and the diagonal matrix as given in equation (33). The identity matrix in the implicit operators is replaced by the product of the similarity-transform matrix and its inverse. A modification is made to the implicit viscous terms by replacing the I_m matrix with an identity matrix so that the transformation matrices may also be factored out of these terms. This modification implicitly adds an additional viscous dissipation term to the pressure. The transformation matrices are now factored out of the implicit operators to give

$$
\begin{aligned}
\mathcal{L}_\xi &= T_\xi[I + \frac{\Delta\tau}{2}J\delta_\xi(\Lambda_\xi - \hat{\gamma}_1)]T_\xi^{-1} \\
\mathcal{L}_\eta &= T_\eta[I + \frac{\Delta\tau}{2}J\delta_\eta(\Lambda_\eta - \hat{\gamma}_2)]T_\eta^{-1} \\
\mathcal{L}_\zeta &= T_\zeta[I + \frac{\Delta\tau}{2}J\delta_\zeta(\Lambda_\zeta - \hat{\gamma}_3)]T_\zeta^{-1}
\end{aligned}
\tag{40}
$$

where the implicit viscous terms are now given by

$$\hat{\gamma}_i = \frac{\nu}{J}\nabla\xi_i \cdot \nabla\xi_i I\delta_{\xi_i}$$

Since the transformation matrices are dependent on the metric quantities, factoring them outside the difference operators introduces an error. No modification has been made to

the right-hand side of the equation and therefore these linearization errors will only effect the convergence path toward the solution, not the steady-state solution.

The implementation of this algorithm instead of the block algorithm will result in a substantial reduction in computational time per iteration because of the decrease in the number of operations performed. Additionally, considerably less memory is required to store the elements on the left-hand side. This additional memory was used to further vectorize the existing code as follows. Since the solution of a tridiagonal block or scalar matrix is recursive, it cannot be vectorized for loops which use the current sweep direction as the inner do-loop index. However, if a large number of these matrices are passed into the inversion routines at once, then vectorization can take place in the 'nonsweep' direction.

Boundary Conditions

An important part of any computer code is the proper implementation of boundary conditions. The code must be capable of handling the several different types of boundaries encountered in numerical simulations which include solid-surface, inflow and outflow, and far-field boundaries.

Solid Surface

At a solid surface boundary, the usual no-slip condition is applied. In general, the grid point adjacent to the surface will be sufficiently close so that constant pressure normal to the surface in the viscous boundary layer can be assumed. For a ζ = constant surface, this can be expressed as

$$\left(\frac{\partial p}{\partial \zeta}\right)_{L=1} = 0 \tag{41}$$

This boundary condition can be implemented either explicitly or implicitly. However, the implicit implementation will enhance the stability of the code. This can be done during the ζ sweep by including the following in the matrix to be inverted.

$$I\Delta D_{j,k,1} + \hat{c}\Delta D_{j,k,2} = \hat{f} \tag{42}$$

where

$$\hat{c} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \hat{f} = \begin{bmatrix} p_{L=2} - p_{L=1} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Inflow, Outflow and Far-field Conditions

The inflow and outflow boundary conditions for an internal flow problem and the far-field boundary conditions for an external flow problem can be handled in much the same way. The incoming flow for both problems can be set to some appropriate constant as dictated by

the problem. For example, at the inlet to a pipe, the pressure can be set to a constant and the velocity profile can be specified. The conditions at the downstream however, are the most difficult to provide. Simple upwind extrapolation is not well-posed. The best convergence rate is obtained if global mass is conserved. So to ensure the best results, the velocities and pressure are first updated using a second-order upwind extrapolation. For an exit at L = LMAX, this is

$$Q^{\tilde{n}}_{lmax} = \frac{Q^{n+1}_{lmax-1}\left(\frac{\Delta z_2}{\Delta z_1}\right) - Q^{n+1}_{lmax-2}}{\frac{\Delta z_2}{\Delta z_1} - 1} \tag{43}$$

where

$$\Delta z_1 = z_{lmax} - z_{lmax-1}$$
$$\Delta z_2 = z_{lmax} - z_{lmax-2}$$

Then these extrapolated velocities are integrated over the exit boundary to obtain the outlet mass flux

$$\dot{m}_{out} = \int_{exit} \vec{V}^{\tilde{n}} \cdot d\hat{a}. \tag{44}$$

Then the velocity components are weighted by the mass flux ratio to conserve global mass

$$\vec{V}^{n+1} = \frac{\dot{m}_{in}}{\dot{m}_{out}} \vec{V}^{\tilde{n}} \tag{45}$$

If nothing further is done to update the boundary pressure, this procedure can lead to discontinuities in the pressure because momentum is not being conserved. A method of weighting the pressure by a momentum correction was presented by Chang et al (ref. 13). They use the following relation to obtain a pressure which corresponds to the mass-weighted velocities

$$p^{n+1} = p^{\tilde{n}} - \frac{1}{\zeta_z}\left[(wW)^{n+1} - (wW)^{\tilde{n}}\right] + \frac{\nu}{\zeta_z}(\nabla\zeta \cdot \nabla\zeta)\left[\left(\frac{\partial w}{\partial \zeta}\right)^{n+1} - \left(\frac{\partial w}{\partial \zeta}\right)^{\tilde{n}}\right] \tag{46}$$

where W is the contravariant velocity. In obtaining this formula, it has been assumed that the streamlines near the exit plane are nearly straight. Any appreciable deviation will cause a discontinuity in the pressure and may lead to an instability. To avoid this, a momentum-weighted pressure was used. This was obtained by integrating the momentum-corrected pressure p^{n+1} and the extrapolated pressure $p^{\tilde{n}}$ across the exit

$$\begin{aligned} I^{n+1}_p &= \int_{exit} p^{n+1} d\hat{a} \\ I^{\tilde{n}}_p &= \int_{exit} p^{\tilde{n}} d\hat{a}. \end{aligned} \tag{47}$$

The final outlet pressure is then taken to be

$$p^{n+1} = \left(\frac{I^{n+1}_p}{I^{\tilde{n}}_p}\right) p^{\tilde{n}} \tag{48}$$

Using these downstream boundary conditions, global conservation of mass and momentum are ensured and the scheme will not introduce instabilities into the flow field.

III. PSEUDOCOMPRESSIBILITY

In an incompressible flow, a disturbance in the pressure causes waves which travel with infinite speed. Introduction of pseudocompressibility causes waves of finite speed, where the magnitude of the speed depends upon the pseudocompressibility constant β. In a true incompressible flow, the pressure field is affected instantaneously by a disturbance in the flow, but with pseudocompressibility, there will be a time lag between the flow disturbance and its effect on the pressure field. In viscous flows, the behavior of the boundary layer is very sensitive to the streamwise pressure gradient, especially when the boundary layer is separated. If separation is present, a pressure wave traveling with finite speed will cause a change in the local pressure gradient which will affect the location of the flow separation. This change in separated flow will feed back to the pressure field, possibly preventing convergence to a steady state.

Wave-Vorticity Interactions

To understand the physical phenomenon of wave propagation and its interaction with viscous effect, a simplified one-dimensional form of the governing equation is anlyzed.

$$\frac{1}{\beta}\frac{\partial p}{\partial t} + \frac{\partial u}{\partial x} = 0 \tag{49}$$

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} = -\frac{\partial p}{\partial x} + \tau_w \tag{50}$$

where τ_w represents wall shear stress. The pseudospeed of sound for this system of equations is given by

$$c = \sqrt{u^2 + \beta}$$

and the pseudo-Mach number can be defined

$$M = \frac{u}{c} = \frac{u}{\sqrt{u^2 + \beta}} \tag{51}$$

which is always less than one for any $\beta > 0$. This equations shows that the present formulation will not generate shocks and also that the solution eventually approaches that of incompressible flows (see ref. 4 for more detail). In three-dimensional applications, u can be considered to be the average velocity in the primary flow direction.

To investigate the coupling effect between the pressure wave and vorticity, consider a channel of length L. The time it takes an upstream propagating wave to travel the distance of the computational domain of length L is given by the dimensionless time scale,

$$T_L = \frac{L/x_{ref}}{c - u} \tag{52}$$

The spreading of the viscous effect can be estimated by the boundary-layer growth rate. Therefore, for laminar flows, the dimensionless time scale for the viscous effect to spread over a thickness, δ, is approximately given by

$$T_\delta \approx \frac{Re}{4}\frac{\delta^2}{x_{ref}} \tag{53}$$

To recover the incompressible phenomena, the physics requires that the pressure wave propagate much faster than the spreading of the viscous effect, namely,

$$T_L << T_\delta \tag{54}$$

From this physical reasoning, a general criterion for the lower bound of β is obtained given as follows:

$$\beta >> \left[1 + \frac{4}{Re}\left(\frac{x_{ref}}{x_\delta}\right)^2\left(\frac{x_L}{x_{ref}}\right)\right]^2 - 1 \tag{55}$$

where x_δ and x_L are the characteristic lengths that the vorticity and the pressure waves have to propagate during a given time span. For a channel flow simulation, x_L is equal to the length of the channel and x_δ is half the channel width. The computational experiments showed this dependency of the pseudocompressibility parameter on the Reynolds number, as well as on the characteristic lengths of the geometry (see ref. 8).

For the near field in external flow simulations where pressure waves propagate out to infinity, the range of acceptable β is less restrictive. The pressure buildup in the near-field region is usually temporary, even if β is not optimum. However, for internal flows, local pressure buildup can be serious when β is not properly chosen.

The total number of iterations required for the wave to travel downstream and back upstream, a total distance of $2L$, must be considered. Recalling that the downstream or upstream travelling waves propagate with speed $(c+u)$ or $(c-u)$ respectively, the total time required for one round trip, τ_1, can be written as

$$\tau_1 = \frac{\sqrt{u^2+\beta}}{\beta}2L \tag{56a}$$

If a constant time step is used throughout the computation, the total number of iterations, N_1, for one round trip is

$$N_1 = \frac{\tau_1}{\Delta\tau} \tag{56b}$$

where $\Delta\tau$ is the increment of dimensionless time. For convergence, enough computing time, τ, has to elapse to permit the wave to make at least one round trip.

This analysis provides a lower bound for the choice of β. For the upper bound of β, the error introduced by the approximate factorization will be considered. Recalling equation (29), the form of the error is

$$E = \left(\frac{\Delta\tau}{2} J^{n+1}\right)^2 \left[\delta_\xi(\hat{A}^n - \Gamma_1)\delta_\eta(\hat{B}^n - \Gamma) + \ldots\right] \tag{57}$$

This term must be kept smaller than the original terms in the equation. If only the terms which contain β are included, this restriction can be expressed as

$$\frac{\Delta\tau}{2} J^{n+1}\delta_{\xi_i}\hat{A}_i^n \delta_{\xi_j}\hat{A}_j^n < \delta_{\xi_i}\hat{A}_i^n, \quad i \neq j$$

or

$$\frac{\Delta\tau}{2} J^{n+1}\delta_{\xi_j}\hat{A}_j^n < 1 \tag{58}$$

Recalling the expression for $\hat{A}_i^n$ given by equation (24), the terms contain β give the following

$$\frac{\Delta\tau}{2}\beta\delta_{\xi_j}\left(\frac{\partial\xi_j}{\partial x_i}\right) < 1 \tag{59}$$

The term to the right of β in this inequality is essentially the change in $\frac{1}{\Delta x_i}$ in either the $\xi, \eta,$ or ζ direction. An estimate of the order of magnitude of this term for the grids generally used is given by

$$O\left[\delta_{\xi_j}\left(\frac{\partial\xi_j}{\partial x_i}\right)\right] \approx 2 \tag{60}$$

which puts the restriction on β

$$O(\beta\Delta\tau) < 1 \tag{61}$$

For most problems, the restrictions for β given by equations (56) and (61) are satisfied with a value for β in the range from 1 to 10.

Effects of Smoothing on Accuracy

The explicit smoothing has a large effect on the convergence and accuracy of the pseudo-compressibility method. In particular, the explicit smoothing on the pressure can affect whether the solution converges to an incompressible flow field. Chang and Kwak (ref. 4) showed that the pseudopressure waves decay exponentially with time and vanish as the solution converges; thus the change in pressure with time approaches zero. When there is no explicit smoothing added to the equation, the divergence of the velocity field identically approaches zero. However, when explicit smoothing is included, as the change in pressure approaches zero, the divergence of velocity approaches

$$\frac{\partial u_i}{\partial x_i} \rightarrow -\frac{\epsilon_e(1)}{\beta\Delta\tau}\left[(\nabla_\xi\Delta_\xi)^2 + (\nabla_\eta\Delta_\eta)^2 + (\nabla_\zeta\Delta_\zeta)^2\right] p \tag{62}$$

where $\epsilon_e(1)$ is the explicit smoothing parameter for the pressure. If ϵ_e is scaled by h, i.e., $\epsilon_e = \Delta\tau\varepsilon_e$, equation (62) becomes independent of time step. This term can deteriorate the conservation of mass depending on the magnitude of β and the local pressure gradient. When the pressure gradients become substantial, as in the case when a region of separation is present, the smoothing term does not approach zero so the divergence of the velocity field is contaminated. In this situation, mesh refinement usually does not help in reducing the magnitude of this term. To alleviate this problem, it may become necessary to decrease the smoothing coefficient in the continuity equation after the solution is almost converged.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-100012		

4. Title and Subtitle

INS3D–An Incompressible Navier-Stokes Code in Generalized Three-Dimensional Coordinates

5. Report Date

November 1987

6. Performing Organization Code

7. Author(s)

S. E. Rogers (Sterling Federal Systems, Palo Alto, CA),
D. Kwak, and J. L. C. Chang (Rocketdyne Division,
Rockwell International, Canoga Park, CA)

8. Performing Organization Report No.

A-87293

10. Work Unit No.

505-60

9. Performing Organization Name and Address

Ames Research Center
Moffett Field, CA 94035

11. Contract or Grant No.

13. Type of Report and Period Covered

Technical Memorandum

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, DC 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes

Point of Contact: Stuart Rogers, Ames Research Center, MS 258-1, Moffett Field, CA 94035
(415) 694-4481 or FTS 464-4481

16. Abstract

The operation of the INS3D code, which computes steady-state solutions to the incompressible Navier-Stokes equations, is described. The flow solver utilizes a pseudocompressibility approach combined with an approximate factorization scheme. This manual describes key operating features to orient new users. This includes the organization of the code, description of the input parameters, description of each subroutine, and sample problems. Details for more extended operations, including possible code modifications, are given in the appendix.

17. Key Words (Suggested by Author(s))

Computational fluid dynamics
Incompressible Navier-Stokes
Pseudocompressibility
User's guide

18. Distribution Statement

Unclassified – Unlimited

Subject Category - 64

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	42	A03