NASA Contractor Report 172585

User's Manual For Airfoil Flow Field Computer Code "SRAIR"

Stephen J. Shamroth
Scientific Research Associates, Inc.
Glastonbury, CT 06033

PREPARED FOR NASA LANGLEY RESEARCH CENTER
CONTRACT NAS1–15214

JUNE 1985

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

TABLE OF CONTENTS

	Page
PROBLEM DEFINITION	1
METHOD OF SOLUTION	1
Mean Flow Equations	1
The Turbulence Model	2
Numerical Procedure	4
Boundary Conditions	5
Artificial Dissipation	6
USER'S INSTRUCTION	7
Flow of the Program	7
Flow of the Program-Input Routines	8
Flow of the Program-Execution Routines	9
Flow of the Program-Output Routines	11
Description of Active Routines	11
LIST OF MAJOR VARIABLES	20
Logical File Units	25
Initial Start/Restart - Steady Calculations	27
Convergence and Run Times	28
Further Comments on Run Protocol	29
IMPLEMENTATION INSTRUCTIONS	31
Control Stream	31
Input	33
Output	40
ENVIRONMENTAL CHARACTERISTICS	42
REFERENCES	43
APPENDIX	45
FIGURES	50

PROBLEM DEFINITION

Program SRAIR has been developed to calculate the flow field about an isolated airfoil. Both flow fields in which the airfoil is steady and oscillating sinusoidally in pitch are considered. The code has been successfully demonstrated for a number of airfoil calculations (e.g. Refs. 1-6). Details of the analysis, numerical method, boundary conditions, etc. are given in Ref. 6

METHOD OF SOLUTION

Mean Flow Equations

The procedure solves the compressible, time-dependent Navier-Stokes equation in conjunction with a mixing length type turbulence model. The form of the equations expressed in the more common coordinate systems can be found in standard fluid dynamic texts. The present approach transforms the Cartesian coordinates (x,y) to a new set of general coordinates (ξ,η) where

$$\xi = \xi(x,y,t)$$

$$\eta = \eta(x,y,t) \tag{1}$$

$$\tau = t$$

The equations can then be expressed as

$$\frac{\partial W}{\partial \tau} + \xi_t \frac{\partial W}{\partial \xi} + \xi_x \frac{\partial F}{\partial \xi} + \xi_y \frac{\partial G}{\partial \xi} + \eta_t \frac{\partial W}{\partial \eta} + \eta_x \frac{\partial F}{\partial \eta} + \eta_y \frac{\partial G}{\partial \eta} = \frac{1}{Re}\left[\xi_x \frac{\partial F_1}{\partial \xi} + \eta_x \frac{\partial F_1}{\partial \eta} + \xi_y \frac{\partial G_1}{\partial \xi} + \eta_y \frac{\partial G_1}{\partial \eta} \right] \tag{2a}$$

where

$$D = \xi_x \eta_y - \xi_y \eta_x$$

$$W = \begin{pmatrix} \rho \\ \rho u \\ \rho v \end{pmatrix}, \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \end{pmatrix}, \quad G = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \end{pmatrix} \quad F_1 = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \end{pmatrix} \quad G_1 = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \end{pmatrix} \tag{2b}$$

A final item related to the choice of equations is the choice of dependent variables. In the present approach the density and velocity components are used as dependent variables. The energy equation is replaced by an assumption of constant total temperature thus leading to a relation between pressure, density and velocity.

$$p = \rho R\left(T^{\circ} - \frac{u^2 + w^2}{2C_p}\right) \tag{3}$$

where R is the gas constant, T° is total temperature and C_p is specific heat.

It should be noted that the energy equation can be solved with the momenta and continuity equations at the cost of adding an additional governing equation which increases computer run time. Calculations of this type in transonic cascades which include comparison with heat transfer data have been made by Weinberg, Yang, Shamroth and McDonald (Ref. 7) using a cascade version of the present code. For steady airfoil flow fields this assumption is reasonable. For unsteady flow, it represents an approximation as can be noted from examination of the unsteady total temperature equation. However, as discussed in Ref. (4), for the cases considered here the assumption of constant total temperature should be valid.

The Turbulence Model

Since the present effort is concerned with high Reynolds number turbulent flows, it is necessary to specify a turbulence model. The present code utilizes a mixing length model. The mixing length model assumes the existence of a mixing length, ℓ, and then relates an eddy viscosity, μ_T, to the mixing length by

$$\mu_T = \rho \ell^2 \left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j}\right]^{1/2} \tag{4}$$

For flow regions upstream of the leading edge where the flow is attached, mixing length is determined by the usual boundary layer formulation

$$\ell = \kappa y D \qquad \ell \leq \ell_{max} \tag{5}$$

where κ is the von-Karman constant, D is a sublayer damping factor and ℓ_{max} is taken 0.09 δ where δ is the boundary layer thickness. The damping factor, D, has for the most part been utilized as the van Driest damping factor

$$D = (1 - e^{-y^+/27}) \tag{6}$$

where y^+ is the dimensionless coordinate normal to the wall, yu_t/ν.

When the mixing length formulation is used in a boundary layer environment, δ is usually taken as the location where $u/u_e = 0.99$. However, this definition assumes the existence of an outer portion of the flow where u_e is independent of distance from the wall and assumes that the location where u_e becomes independent of distance from the wall marks the end of the viscous region. In an airfoil Navier-Stokes calculation no such clear flow division occurs as u approaches the upstream velocity, u_∞, as distance from the wall increases. Therefore, the boundary layer thickness, δ, is set by first determining u_{max}, the maximum velocity at each given streamwise station, and then setting δ by

$$\delta = 2.0 y_{(u/u_{max} = k_1)} \tag{7}$$

i.e., δ is taken as twice the distance from the wall to the location where $u/u_{max} = k_1$ where k_1 is set to 0.90. If the flow is separated, then a minimum mixing length is set by

$$\ell_{min} = 0.1\, hD \tag{8}$$

where h is the local height of the separated region. In the wake the mixing length is made proportional to the wake thickness, δ, and a linear

growth of δ with distance is assumed based upon classical free jet boundary growth (e.g. Ref. (8)). With this assumption

$$\delta = (\delta_{ps} + \delta_{ss}) + 0.2\,(X - X_{TE}) \tag{9}$$

where δ_{PS} and δ_{SS} are the pressure and suction surface trailing edge boundary layer thicknesses and X_{TE} is the trailing edge location. The mixing length, ℓ, is taken as 0.2δ. The viscosity is blended between regions obtained using the wall formulation for ℓ and the wake formulation for ℓ. Having obtained the turbulent viscosity, μ_T, the turbulent stress, $\overline{-\rho u_i' u_j'}$ is given by

$$-\overline{\rho u_i' u_j'} = \mu_T\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right] \tag{10}$$

Although the mixing length model does not include a transition model, transition can be simulated by specifying a location upstream of which the flow is laminar. This corresponds to forced transition. Even if no forced transition is assumed, the flow in the leading edge region will be laminar as the boundary layer thickness becomes very small in this region.

Numerical Procedure

The numerical procedure used to solve the governing equations is a consistently split linearized block implicit (LBI) scheme originally developed by Briley and McDonald (Ref. 9). A conceptually similar scheme has been developed for two-dimensional MHD problems by Lindemuth and Killeen (Ref. 10). The procedure is discussed in detail in Refs. 9 and 11. The method can be briefly outlined as follows: the governing equations are replaced by an implicit time difference approximation, optionally a backward difference or Crank-Nicolson scheme. Terms involving nonlinearities at the implicit time level are linearized by Taylor expansion in time about the solution at the known time level, and spatial difference approximations are introduced. The result is a system

of multidimensional coupled (but linear) difference equations for the dependent variables at the unknown or implicit time level. To solve these difference equations, the Douglas-Gunn (Ref. 12) procedure for generating alternating direction implicit (ADI) schemes as perturbations of fundamental implicit difference schemes is introduced in its natural extension to systems of partial differential equations. This technique leads to systems of coupled linear difference equations having narrow block-banded matrix structures which can be solved efficiently by standard block-elimination methods.

The method centers around the use of a formal linearization technique adapted for the integration of initial-value problems. The linearization technique, which requires an implicit solution procedure, permits the solution of coupled nonlinear equations in one space dimension (to the requisite degree of accuracy) by a one-step noniterative scheme. Since no iteration is required to compute the solution for a single time step, and since only moderate effort is required for solution of the implicit difference equations, the method is computationally efficient; this efficiency is retained for multidimensional problems by using what might be termed block ADI techniques. The method is also economical in terms of computer storage, and its present form requiring only two time levels of storage for each dependent variable. Furthermore, the block ADI technique reduces multidimensional problems to sequences of calculations which are one-dimensional in the sense that easily solved narrow block-banded matrices associated with one-dimensional rows of grid points are produced. A more detailed discussion of the solution procedure is discussed by Briley, Buggeln and McDonald (Ref. 13) and is given in the Appendix.

Boundary Conditions

An important component of the airfoil analysis concerns specification of boundary conditions. The present analysis considers 'C' grid coordinate systems and requires boundary conditions to be set along the lines, $\xi = \xi_{min}$, $\xi = \xi_{max}$, $\eta = \eta_{min}$ and $\eta = \eta_{max}$. With the coordinate system sketched in Fig. 1, $\xi = \xi_{min}$ (line AA') and $\xi = \xi_{max}$ (line DD') are downstream boundaries. In the early work done under this

effort derivatives were set to zero at this boundary and function conditions specified on the remainder of the outer boundary. On the airfoil surface no-slip conditions are used in conjunction with an inviscid momentum equation (which for no motion and no heat transfer reduced to zero density gradient) as boundary conditions. More recently the boundary conditions were modified based upon a suggestion by Briley and McDonald (Ref. 14). Following this suggestion, static pressure is specified along with velocity derivatives along the downstream boundaries (lines AA' and DD' of Fig. 1) and along the aft portion of the outer boundary (line segments AB and CD). Total pressure, angle of incidence and the density derivative are specified along the inflow portion of the outer boundary segment BC. This represents the default set of boundary conditions and is the set recommended. Finally, calculations have been made in which tunnel wall boundary conditions are simulated by specifying the flow direction and a full slip condition along AB and CD.

Artificial Dissipation

One major problem to be overcome in calculating high Reynolds number flows using the Navier-Stokes equations is the appearance of spatial oscillations associated with the so-called central difference problem. When spatial derivatives are represented by central differences, high Reynolds number flows can exhibit a saw tooth type oscillation unless some mechanism is added to the equations to suppress their appearance. This dissipation mechanism can be added implicitly to the equations via the spatial difference molecule (e.g. one-sided differencing) or explicitly through addition of a specific term. The present author favors this latter approach for two reasons. First, if a specific artificial dissipation term is added to the equations, it is clear precisely what approximation is being made. Secondly, if a specific term is added to suppress oscillations, the amount of artificial dissipation added to the equations can be easily controlled in magnitude and location so as to add the minimum amount necessary to suppress spatial oscillations.

The present code adds a term of the form $\partial(\nu_{art}\,\partial\phi/\partial z)/\partial z$ to the governing equation where $\phi = \rho$, u, v for the continuity, x-momentum and

y-momentum equations respectively and ν_{art} is determined by

$$\frac{U_z \Delta Z}{\nu + (\nu_{art})_z} \leq \frac{1}{\sigma_z} \tag{11}$$

In the above equation ΔZ is the distance between grid points in a given coordinate direction, U_Z is the velocity in this direction, σ_Z is the artificial dissipation parameter for this direction and ν is the kinematic viscosity. The equation determines ν_{art} with ν_{art} taken as the smallest non-negative value which will satisfy the expression. It should be noted that in two space dimensions each equation contains two artificial disspation terms, one in each coordinate direction. For example, the streamwise momentum equation expressed in Cartesian coordinates would contain the artificial dissipation terms

$$\frac{\partial}{\partial x}\left[(\nu_{art})_x \frac{\partial y}{\partial x}\right] + \frac{\partial}{\partial z}\left[(\nu_{art})_z \frac{\partial y}{\partial z}\right] \tag{12}$$

The parameter σ_Z is set via input. This approach has been used in a variety of calculations, e.g. Refs. (4-6, 15-17).

USER'S INSTRUCTION

Flow of the Program

The program is divided into three major parts which are the input, execution and output sections. These sections are called from PROGRAM DAL as shown in Fig. 2. PROGRAM DAL sets input via a series of DATA STATEMENTS and then calls READA. READA and its associated subroutines set the remaining input data and, if necesary, construct an initial flow field. Control is then returned to DAL which calls EXEC. EXEC and its associated subroutines solve the equations via a time marching procedure. In addition, some program output is written by routines called from EXEC. Finally, after a specified number of time steps have been calculated program restart is written via the call to RESTRT and a final flow field is printed via a call to PRNTF. These operations are now discussed in more detail.

Flow of the Program-Input Routines

The program input routines are called through SUBROUTINE READA which is called from PROGRAM DAL. The flow through this section of the code is sketched in Fig. 3. READA first sets default values for most program input variables and the calls RDINP1 which is an ENTRY POINT in SUBROUTINE READB. RDINP1 first reads NAMELIST READ1 which indicates if the run is an initial run or a restart. If the case being run is being restarted from the results of a previous case, RDINP1 also reads this same input from a restart binary file. RDINP1 then reads namelist input from the input file. In the case of a restart run this card input is only required to change input from that which exists on the restart binary file. Control is then returned to READA and integer variables set at this time include information for disk I/O, information for the number of equations to be solved, the equation coupling procedure and print flags. In addition, various dimensionless groupings which are repeatedly required in the analysis are calculated. The program then calls BVIV which performs the flow field initialization. SUBROUTINE BVIV first calculates integer variables which set the limits of the computational grid and then calculates the required first and second derivative difference weights. If the case being run is a restart case, control returns to READA at this point. If not, BVIV calls FLWFLD which is used to construct an initial flow field.

When constructing an initial flow field, the code passes through SUBROUTINE FLWFLD and SUBROUTINE SPREAD. SUBROUTINE SPREAD constructs a uniform velocity initial flow field with gradual change to no-slip, no through flow conditions at the airfoil surface. After construction of the initial flow field control returns to READA. If the case being considered is a restart case, the initial flow field calculation performed in SPREAD is not required and FLWFLD and SPREAD are not called. Instead RDINP2, an entry point in READB, is called to set the initial flow field. The final operations performed in READA are the writing of NAMELIST data on the OUTPUT file via a call to WRLIST (an ENTRY POINT in SUBROUTINE RDLIST). After writing the required information control is returned to DAL.

In addition to the initialization processes described above certain checks are made in READA to insure consistent input. Most of these checks are satisfied due to default setting of input parameters, however, the

computational grid is currently limited to 6000 grid points and if an attempt is made to use more grid points the run will abort. Other limits which must be observed require the number of grid points in the pseudo-azimuthal (x or ξ) direction to be less than or equal to 150 and the maximum number of grid points in the pseudo-radial (z or η) direction to be less than or equal to 40. The maximum grid run with this specific version of the code has been 141 × 39 and it is recommended that these limits be observed.

Flow of the Program-Execution Routines

After reading input and initializing the flow field (or reading restart data in case of a restart), PROGRAM DAL calls SUBROUTINE EXEC. Subroutine EXEC and ADIC are the main calling programs for the execution portion of the code during which the equations are marched in time via an ADI procedure. A flow diagram of EXEC is presented in Fig. 4. SUBROUTINE EXEC first calls EXTBV which insures that the dependent variables on the boundaries are consistent with the specified boundary conditions. Then, if requested (if IDUMP1=1), the flow field at the initiation of the run is printed. Following this optional print, the program enters the DO 1000 loop which marches the solution in time. TIMGEO is called to calculate metric coefficients in cases where they are not steady: for example, an oscillating airfoil calculation; for cases where they remain constant with time, the geometry is not recalculated. Each pass through the loop advances the solution one time step. SUBROUTINE ADIC is in this loop and it is within ADIC that the equations are solved. After solving the equations, the time step increment is reset and results for the time step are written on the OUTPUT file. When the specified number of time steps have been calculated, control is passed back to DAL.

The main controlling routine in the actual equation solving process is SUBROUTINE ADIC; flow through this program is sketched in Fig. 5. SUBROUTINE ADIC consists of two major DO LOOPS. DO 700 sweeps across the flow field and solves successive LZ constant lines (i.e., ξ-implicit lines such as PQR of Fig. 1) by calling ADICX. This represents the first sweep of the ADI procedure. Within ADICX a check is made to determine if LZ=LZ1; i.e., if the ξ-implicit line is the first line off the inner

surface. When ADICX is called with LZ=LZ1, the equations are solved along two ξ-implicit lines in this one call; the lines solved are the branch cut line emanating from the airfoil trailing edge and the first pseudo-azimuthal line away from the airfoil. If LZ$\neq$LZ1 then only a single ξ-implicit line is solved within the given subroutine call.

If the equations are being solved along the branch cut line, storage is rearranged before proceeding; the equations then are solved. The matrix representing the linearized equations is generated within ADICP1 as shown in Fig. 6. SUBROUTINE ADICP1 first clears the matrix arrays and then calls AMATRX, DOP1 and DOP3. The latter two calls transfer control to ENTRY points within SUBROUTINE DOP. AMATRX loads the matrix arrays with the linearized representation of the time-dependent term; DOP1 and DOP3 load the arrays with the linearized representations of the spatial terms. Upon completing DO LOOP 300 the matrix representing the linearized equations for all interior points along the specific ξ-implicit line which is under consideration is generated. The program then calls GENBC which in turn calls BC to generate the boundary condition representation. The equations are solved in MG3X3 which inverts a block tri-diagonal matrix and the solution is stored for future use. These operations complete the first sweep.

The program then returns control to ADIC which calls SUBROUTINE ADICZI; this latter routine sweeps through the grid solving η-implicit lines one at a time such as ST or FC of Fig. 1 and thus performs the second ADI sweep. This second sweep sketched in Fig. 7 is analogous to this first sweep with one exception. In the second sweep, the calculation is carried through the branch cut in the portion of the flow field aft of the airfoil (LX$\leq$IDBL) such as line ST and the calculation in this region requires storage rearrangement to place the geometry and dependent variables in proper core locations for solution. The previous description of the analysis has termed the coordinates x and y and the Cartesian velocity components u and v. This is in agreement with previous descriptions of the analysis. However, the code is written in terms of x, z and u, w and, therefore, the description of the code will be consistent with this latter nomenclature.

Finally ADIC calls EXTBV which updates the dependent variable array, calculates pressure and temperature, calculates the maximum change of dependent variable over the time step and calculates the turbulent viscosity.

Flow of the Program-Output Routines

The final portion of the SRAIR code to be considered is the program output. This consists of three parts: (i) flowfield data written at each time step, (ii) flowfield data written at the completion of the run and (iii) restart data. The flowfield data written at each time step is obtained via CALL PRNTS where PRNTS is an ENTRY point in SUBROUTINE PRNTA. The dependent variable arrays written at the end of each run are obtained via CALL PRNTF which is also an ENTRY point of PRNTA. Finally, restart data is obtained via CALL RESTRT. The output data is described in detail subsequently.

Description of Active Routines

PROGRAM DAL

PROGRAM DAL is the main program of the SRAIR code. PROGRAM DAL sets variables via data statements and then calls READA which sets the input and if necessary, calls the routines required to construct an initial flow field. The program then calls EXEC which marches the equations a required number of time steps, calls RESTRT which writes restart data on tape and calls PRNTF which prints the final flow field output.

SUBROUTINE ADIC

SUBROUTINE ADIC is the controlling subroutine for the two sweep ADI procedure. The subroutine calls ADICX which is the controlling subroutine for the first ADI sweep, calls ADICZI which is the controlling subroutine for the second ADI sweep and calls EXTBV which performs a variety of operations that include the updating of the dependent variables at the new time step. A pass through ADIC represents a complete time step operation.

SUBROUTINE ADICP1

SUBROUTINE ADICP1 loads the matrix array representing the linearized equations and boundary conditions for a set of coupled equations and calls MG3X3 to invert the matrix. A pass through ADICP1 represents the solution along a single ξ-implicit direction line where ξ is the first sweep direction.

SUBROUTINE ADICP2

SUBROUTINE ADICP2 is analogous to ADICP1 except a pass through ADICP2 represents the solution of a η-implicit line where η is the second sweep direction.

SUBROUTINE ADICX

SUBROUTINE ADICX is the controlling subroutine for the first direction ADI sweep. It is called from ADIC and contains a DO LOOP which ranges over all ξ-implicit lines. Special logic is present for the coordinate system branch cut emanating from the airfoil trailing edge.

SUBROUTINE ADICZI

SUBROUTINE ADICZI is the controlling subroutine for the second direction ADI sweep. It is analogous to ADICX with special logic being required for the η-implicit lines which pass through the coordinate system branch cut.

SUBROUTINE ADIUN1

SUBROUTINE ADIUN1 is analogous to ADICP1 but is for a single equation.

SUBROUTINE ADIUN2

SUBROUTINE ADIUN2 is analogous to ADICP2 but is for a single equation.

SUBROUTINE AMATRX

SUBROUTINE AMATRX is a calling routine for ATIME. It is called from ADICP1, ADICP2, ADIUN1, ADIUN2.

SUBROUTINE ARTVIS1

SUBROUTINE ARTVIS1 calculates artificial viscosity and loads contribution of artificial dissipation terms to linearized difference equations.

SUBROUTINE ATIME

SUBROUTINE ATIME loads contribution of time dependent terms to linearized difference equations. It is called from AMATRX.

SUBROUTINE BC

SUBROUTINE BC is called by GENBC and sets the coefficients representing the linearized boundary condition equations. The type of boundary condition to be set depends upon the value of JEQBC (see Description of Input). The choice of boundary conditions includes function, first derivative and second derivative boundary conditions for ρ, u, and w as well as static pressure and its first derivative and total pressure and its first derivative.

SUBROUTINE BCPM

SUBROUTINE BCPM is used in conjunction with BC in setting static and total pressure boundary conditions. It is called from BC.

SUBROUTINE BVIV

SUBROUTINE BVIV is called from READA to set the limits of the computational domain and the values of the finite difference molecules. It also calls FLWFLD which sets the initial flow field.

SUBROUTINE CONVCT1

SUBROUTINE CONVCT1 which is called from GENEQ1 loads contribution of convective terms to linearized difference equations.

SUBROUTINE COORD

SUBROUTINE COORD is a passive routine which calls INTGEO.

SUBROUTINE CORNER

SUBROUTINE CORNER is called from BVIV and sets up grid point limits for the computational mesh.

SUBROUTINE DELU

SUBROUTINE DELU calculates dilitation terms used in the momentum equations.

SUBROUTINE DIFF1

SUBROUTINE DIFF1 which is called from GENEQ1 loads contribution of diffusive terms to linearized difference equations.

SUBROUTINE DIV

SUBROUTINE DIV calculates velocity divergence.

SUBROUTINE DOP

SUBROUTINE DOP is called from ADICX and ADICZI. It has two ENTRY POINTS; these are DOP1 and DOP3. The linearized representation of the ξ-derivatives and source terms are calculatd in DOP1 and that of the η-derivatives are calculatd in DOP3.

SUBROUTINE EOS

SUBROUTINE EOS is a general equation of state subroutine which calculates pressure and/or temperature.

SUBROUTINE EOSDP

SUBROUTINE EOSDP linearizes the pressure gradient terms in the momentum equations and loads their contribution to the linearized difference equation.

SUBROUTINE EOSUP1

SUBROUTINE EOSUP1 updates pressure and temperature. It is called from TEMPN.

SUBROUTINE EXEC

SUBROUTINE EXEC is the main control routine of the execution (or time-marching) portion of the program. EXEC is called from DAL and contains a DO LOOP which marches through a specified number of time steps. Within this loop the program calls ADIC. ADIC marches the solution through a single time step. The program then calls PRNTS which prints the maximum change in dependent variables over the time step. Finally, the program sets the magnitude of the next time increment and if an abort flag has been set (KILL≠0), the program prints the flow field and terminates.

SUBROUTINE EXTBV

SUBROUTINE EXTBV is called from ADIC after the two sweeps required for an ADI solution have been completed. The routine updates boundary values on the first sweep boundary points to make the boundary values consistent with the second sweep solution. This operation is performed in SETBV. The subroutine then updates the dependent variable array, calculates the pressure and temperature field and, if necessary, calls VISCOS to calculate turbulent viscosity.

SUBROUTINE FLWFLD

SUBROUTINE FLWFLD calls SPREAD to set up the initial flowfield.

SUBROUTINE GASP

SUBROUTINE GASP sets molecular viscosity

SUBROUTINE GAUSS

SUBROUTINE GAUSS solves the matrix equation Ax=B for x.

SUBROUTINE GENBC

SUBROUTINE GENBC is called from ADICX and ADICZI to set the boundary condition equations for a set of coupled equations. The subroutine calls BC to determine the boundary equation coefficients and then loads the coefficients in the coefficient matrix array.

SUBROUTINE GENEQ1

SUBROUTINE GENEQ1 calls ARTVIS1, CONVCT1, DIFF1, etc. to form the algebraic difference equations required for matrix inversion.

SUBROUTINE GENUBC

SUBROUTINE GENUBC sets boundary conditions for a single equation.

SUBROUTINE MG3X3

SUBROUTINE MG3X3 inverts a block tri-diagonal matrix.

SUBROUTINE INTGEO

SUBROUTINE INTGEO is the geometry initialization subroutine which is called at the beginning of a run to set geometry data in conjunction with TIMGEO.

SUBROUTINE MGAUSF

SUBROUTINE MGAUSF inverts a block tridiagonal matrix. The maximum block size allowable is 4 x 4.

SUBROUTINE MGERR

SUBROUTINE MGERR calculates error check for MGAUSF.

SUBROUTINE NORMD

SUBROUTINE NORMD is an input normalization routine.

SUBROUTINE PRGEO

SUBROUTINE PRGEO prints out the geometric data.

SUBROUTINE PRINT1

SUBROUTINE PRINT1 is called from RESULT and prints a specified dependent variable array.

SUBROUTINE PRNTA

SUBROUTINE PRNTA has several ENTRY points; these are PRNTA, PRNTB, PRNTF and PRNTS. The first three are used to print the dependent variable arrays by calling RESULT. The last ENTRY point prints the change in variables across a time step.

SUBROUTINE QUICK

SUBROUTINE QUICK is a matrix invertor used in conjunction with MGAUSF, MG3X3 and SETBV.

SUBROUTINE RDLIST

SUBROUTINE RDLIST reads NAMELIST input and writes this input on an output file. In addition RDLIST is used to write restart files.

SUBROUTINE READA

SUBROUTINE READA is the main controlling routine for the input portion of the program; it is called from DAL. READA first sets default values for a variety of input variables and then calls RDINP1, an ENTRY point in READB. During this call the NAMELIST input are read in RDLIST. Following the reading of NAMELIST input, various flags indicating the number of equations to be solved, how equations are to be coupled, etc. are set and dimensionless groupings and reference quantities are calculated. The subroutine then calls BVIV which has been described previously. Finally, if the case being run is a restart of a previous case, READA then calls RDINP2, a second ENTRY point in READB which reads the required restart data arrays.

SUBROUTINE READB

SUBROUTINE READB has two ENTRY points. ENTRY point RDINP1 reads NAMELIST input from cards and/or a binary restart file. ENTRY point RDINP2 reads other required restart data from a restart file.

SUBROUTIN RESTRT

SUBROUTINE RESTRT writes the restart files.

SUBROUTINE RESULT

SUBROUTINE RESULT is called from PRNTA and calls PRINT1. It is part of the sequence used to write the dependent variable arrays.

SUBROUTINE SETBV

SUBROUTINE SETBV is used to set boundary conditions on the boundary points of the ξ-direction implicit lines after the second ADI sweep has been made. Use of this subroutine updates these boundary points so that they are compatable with second sweep results for interior points.

SUBROUTINE SETBVI

SUBROUTINE SETBVI is called to insure that the initial flow field is consistent with the specified boundary conditions. Consistency is obtained by changing the values of the dependent variables on the computational grid boundary lines if necessary.

SUBROUTINE SOURCE

SUBROUTINE SOURCE loads contributions of source type terms to linearized difference equations.

SUBROUTINE SPREAD

SUBROUTINE SPREAD is part of the flow initialization procedure. If a case is not a restart case, it is called from FLWFLD and calculates velocity components at each grid point. It then calculates density and turbulence energy.

SUBROUTINE SSTST

SUBROUTINE SSTST sweeps through the flow field and calculates the maximum change in u, w and ρ (the two velocity components and density) across each time step.

SUBROUTINE VISCOS

SUBROUTINE VISCOS is called from EXTBV and is used to calcuate turbulent viscosity.

SUBROUTINE TEMPN

SUBROUTINE TEMPN is a calling routine which calls EOSUP1 to calculate temperature and/or pressure.

SUBROUTINE TIMGEO

SUBROUTINE TIMGEO reads coordinate data on a line by line basis and creates required metric coefficients and Jacobian.

SUBROUTINE WRPLOTB

SUBROUTINE WRPLOTB writes plot files.

LIST OF MAJOR VARIABLES

FORTRAN SYMBOL	COMMON BLOCK	DESCRIPTION
AC(I,J,K)	BLK1	DEPENDENT VARIABLE ARRAY
ACG(J,J)	BLK1	GEOMETRY DATA ARRAY
AN(I,J)	BLKM	ARRAY STORING TIME TERM LINEARIZED COEFFICIENTS
APR(I,J)	PRNT	PRINT OUTOUT ARRAY
AVISC(I,J)	MISC2	ARTIFICIAL DISSIPATION PARAMETER
C(I,J,K)	BLKM	COUPLED MATRIX ARRAY STORGAGE
CLENG	CREF	REFERENCE LENGTH
CMACH	MISC2	REFERENCE MACH NUMBER
D	VARNO	INDEX FOR DIVERGENCE
D1(I,J,K)	BLKM	ARRAY STORING FIRST SWEEP LINEARIZED COEFFICIENTS
D2(I,J,K)	BLKM	ARRAY STORING SECOND SWEEP LINEARIZED COEFFICIENTS
D3(I,J,K)	BLKM	ARRAY STORING THIRD SWEEP LINEARIZED COEFFICIENTS
DENSR	CREF	REFERENCE DENSITY
DFW(I,J,K)	AD17	DIFFERENCE WEIGHT ARRAY
DIM1	NOND	INVERSE REYNOLDS NUMBER
DIM2	NOND	REFERENCE PRESSURE/REFERENCE DYNAMIC HEAD
DIM3	NOND	REFERENCE PRESSURE/(REFERENCE DENSITY * REFERENCE ENTHALPY)
DIM4	NOND	$1.0/(REY * P_r)$
DIM12	NOND	2.0 * DIM1

FORTRAN SYMBOL	COMMON BLOCK	DESCRIPTION
DS	VARNO	INDEX FOR DISSIPATION
DT	MISC2	TIME STEP
DTCON	MISC2	INVERSE STEP
DTMAX	MISC2	MAXIMUM ALLOWABLE TIME STEP
DTMIN	MISC2	MINIMUM ALLOWABLE TIME STEP
E(I,J,K)	BLKM	COUPLED MATRIX ARRAY STORAGE
GRID(I)	GTRAN	GRID DISTRIBUTION PARAMETER
H	VARNO	INDEX FOR ENTHALPY
I1	MGAUS	LOWER LIMIT FOR MATRIX INVERSION
IADI	ADI1	ADI SWEEP NUMBER
IBC	ADI1	BOUNDARY CONDITION BOUNDARY PARAMETER
IDT	MISC2	TIME STEP INDEX
IDTADJ	MISC2	TIME STEP CONTROL PARAMTER
IDUMP1	OUTA	PARAMETER CONTROLLING INITIAL STATION PRINT
IEQ	ADI1	EQUATION NUMBER
IGPRT(I)	GEO1	GEOMETRY PRINT CONTROL
IL	MGAUS	UPPER LIMIT FOR MATRIX INVERSION
IPRINT	MISC2	PRINT INTERVAL PARAMETER
IREST	MISC2	RESTART READ CONTROL PARAMETER
IVARPR(I)	MISC2	PRINT PARAMETER
JADI	ADI1	ADI SWEEP PARAMETER
JEQBC(I,J,K)	ADI1	BOUNDARY CONDITION TYPE PARAMETER

FORTRAN SYMBOL	COMMON BLOCK	DESCRIPTION
JX	ADI2	DIRECTION-1 GRID POINT INDEX
KZ	ADI2	DIRECTION-3 GRID POINT INDEX
LX	ADI2	DIRECTION-1 GRID POINT INDEX
LX1	ADI3	FIRST DIRECTION-1 INTERIOR POINT
LX2	ADI3	LAST DIRECTION-1 INTERIOR POINT
LY	AD12	DIRECTION-2 GRID POINT INDEX
LY1	ADI3	FIRST DIRECTION-2 INTERIOR POINT
LY2	ADI3	LAST DIRECTION-2 INTERIOR POINT
LZ	ADI2	DIRECTION-3 GRID POINT INDEX
LZ1	ADI3	FIRST DIRECTION-3 INTERIOR POINT
LZ2	ADI3	LAST DIRECTION-3 INTERIOR POINT
MEQS	ADI1	NUMBER OF EQUATIONS TO BE SOLVED
NT	MISC2	NUMBER OF TIME STEPS TO BE RUN
NUMDX	MISC2	NUMBER OF INTERIOR DIRECTION-1 POINTS
NUMDY	MISC2	NUMBER OF INTERIOR DIRECTION-2 POINTS
NUMDZ	MISC2	NUMBER OF INTERIOR DIRECTION-3 POINTS
NX1	ADI4	FIRST GRID POINT - DIRECTION-1
NX2	ADI4	LAST GRID POINT - DIRECTION-1
NY1	ADI4	FIRST GRID POINT - DIRECTION-2
NY2	ADI4	LAST GRID POINT - DIRECTION-2
NZ1	ADI4	FIRST GRID POINT - DIRECTION-3
NZ2	ADI4	LAST GRID POINT - DIRECTION-3

FORTRAN SYMBOL	COMMON BLOCK	DESCRIPTION
P	VARNO	INDEX FOR PRESSURE
PCNT1	MISC2	TIME STEP CONTROL PARAMETER
PCNT2	MISC2	TIME STEP CONTROL PARAMETER
PREF	CREF	REFERENCE PRESSURE
PRNDL	CREF	PRANDTL NUMBER
PTOT	BCCON	TOTAL PRESSURE
R	VARNO	INDEX FOR DENSITY
REY	CREF	REYNOLDS NUMBER
SN(I)	BLKM	ARRAY STORING SOURCE TERM LINEARIZED COEFFICIENT
SSTEST	MISC2	MAXIMUM CHANGE IN VARIABLE ACROSS TIME STEP
T	VARNO	INDEX FOR TEMPERATURE
TAUW	TURB	WALL SHEAR
TREF	CREF	REFERENCE TEMPERATURE
TTIME	MISC2	CUMULATIVE TIME
TTOT	BCCON	TOTAL TEMPERATURE
U	VARNO	INDEX FOR DIRECTION-1 VELOCITY
USTAR	TURB	DIMENSIONLESS VELOCITY
V	VARNO	INDEX FOR DIRECTION-2 VELOCITY
VISCL	TURB	LAMINAR REFERENCE VISCOSITY
VISCR	CREF	REFERENCE VISCOSITY
VS	VARNO	INDEX FOR VISCOSITY
W	VARNO	INDEX FOR DIRECTION-3 VELOCITY

FORTRAN SYMBOL	COMMON BLOCK	DESCRIPTION
WREF	CREF	REFERENCE VELOCITY
XGMAX(I)	GRID1	MAXIMUM COORDINATE VALUE
XGMIN(I)	GRID1	MINIMUM COORDINATE VALUE

Logical File Units

The SRAIR computer code utilizes several logical file units during the execution of the run. In general, the file units are referenced via FORTRAN names allowing easy change of logical file unit numbers. The files used are:

FORTRAN Name	Default Unit Number	Description
KTAPE	1	Plot File
MINP	5	Input File
MPRT	6	Output File
INTAPE	9	Restart Input File
IOTAPE	10	Restart Output File
INTAP1	19	Restart Input File
IOTAP1	20	Restart Output File
	21	Geometry Input File
	2	Scratch File

File MINP is an input file used for namelist input The card namelist input must be copied to TAPE5 via JCL instructions prior to execution. TAPE21 contains the required coordinate locations for each grid point. If computer code CRDSRA is used to generate the coordinates, CRDSRA writes

the coordinates on TAPE9 and as discussed in the CRDSRA User's Guide, this file must be permanently stored to be used as input for SRAIR. Generation of coordinates by other techniques are discussed in the next subsection.

File KTAPE is used for plot information output. Plot information is written for seven items on a line by line basis. The items written are x-coordinate, z-coordinate, u-velocity, w-velocity, density, pressure coefficient and viscosity. Each item is written as formatted output via

```
FORMAT (I4, 1X, 10E12.5)
```

where the first variable is the LZ line number and the next ten are the items in question. It should be noted that LZ lines are lines such as PQR of Fig. 1. The first line is A'ED' which includes the branch cut, the airfoil surface and the branch cut. The last line is AFD. If each line contains NZ2 grid points then NZ2/10 records per line are required where NZ2/10 is rounded up to the next integer. The variables u, w, ρ, and μ are normalized by reference values; whereas the values for p are written as P/Pref - 1.0.

Output is written on file MPRT. If restart information is sought, it is written on IOTAPE and IOTAP1. IOTAPE contains namelist items; IOTAP1 contains the dependent variable arrays. Restart information required for starting in a restart mode is obtained from INTAPE and INTAP1 where INTAPE corresponds to the previously written IOTAPE and INTAP1 corresponds to the previously written IOTAP1.

Coordinate Input - TAPE21

As discussed above coordinate input is required and can be obtained from computer code CRDSRA. However, if other coordinate generation procedures are available, they can be used in conjunction with SRAIR. The coordinate data is read into the program in SUBROUTINE TIMGEO prior to the DO 238 loop. The coordinates are read for each z-line via

```
READ(21)  (XLOCA(MX),  ZLOCA(MX),  MX=1,NX2)
```

where XLOCA and ZLOCA and the X and Z locations of grid points. NZ2 reads must be made since the number of LZ lines is NZ2. The airfoil chord is unity in length with the leading edge point at X = 0.0, Z = 0.0 and the trailing edge at X = 1.0, Z = 0.0. The coordinate system is a 'C' type grid as shown in Fig. 1. The first LZ line is line A'ED' and the last LZ line is AFD. In regard to the coordinate system itself, it is important that metric coefficients and Jacobians vary smoothly from grid point to grid point and the coordinate lines and metric data must be continuous across the trailing edge grid branch cut.

Initial Start/Restart - Steady Calculations

Since it is often desirable to make a complete calculation in a series of runs, the code can be run either as an initial start (IREST=0 input) or a restart from a previous solution (IREST=1). When run as an initial start all input occurs via NAMELIST on cards. The input is discussed in detail in a subsequent section of this guide.

In regard to run strategy, it is suggested that for calculations in which the airfoil is steady the initial and minimum time steps, DT and DTMIN, be set to 1.0 the maximum time step DTMAX be set to 10.0 and ITCALC be set to 5. The artificial dissipation parameter, σ, which is termed AVISC should be set to 0.1 and turbulent flow assumed over the entire airfoil. After running for approximately sixty time steps obtain a restart and run an additional 40 time steps with $0.05 < \text{AVISC} < .025$ and with a set transition location. Previous experience indicates this will lead to converged solutions.

In regard to calculations for an oscillating airfoil between α_{min} and α_{max}, experience has shown that this calculation should be initiated by obtaining a converged steady solution at $\alpha = \alpha_{min}$. The oscillating solution should be started from this converged solution with a restart. Oscillating airfoil solutions should not be started from an initial 'cold start' run. In starting the solution since the incidence is given by

$$\alpha = \alpha_o + \frac{\Delta\alpha}{2} \{1 - \cos[w(t-t_o)]\}$$

the variable, $t-t_o$ should be set to zero at the initiation of the unsteady run. This is obtained by setting TTIME=0.0. In running oscillating airfoil calculations, physical transients must be resolved which requires marching in physical time; i.e., matrix preconditioning for rapid convergence to steady state cannot be used. This requires setting ITCALC=0. The minimum time step DTMIN should be set to 0.01 and the maximum time step, DTMAX, to 0.10. The initial time step, DT, should be set to DTMIN.

In performing a restart NAMELIST READ1 is first read from cards to determine if the run is a restart. The remaining data is then read from the restart file, INTAP1. This contains values of the parameters at the end of the restart run. Only if these values are inappropriate need they be overwritten by the card NAMELIST input.

Convergence and Run Times

When a steady flow is sought via a time marching technique, the question arises as to when convergence is obtained. In considering this question, several factors must be taken into account. First of all, not all flows reach a steady state. For example, airfoils at high incidence which shed vortices or airfoil flows in which a shock wave is present may never become truly steady. In the former case, vortices are shed in some quasi-periodic manner and the unsteadiness has a large time scale. In the latter case, the shock position may move leading to an unsteadiness with a small time scale. Obviously, in these cases no steady flow solution is guaranteed. Secondly, the numerical technique used may hinder complete convergence. For example, in the present approach the turbulent viscosity is lagged by one step in time and this interaction between the viscosity evaluation and the mean flow calculation may hinder or even prevent complete convergence.

In general several items are monitored to determine convergence. First of all the calculated surface pressure distribution, the location of any separation points and the velocity field in the vicinity of the airfoil. In addition, the output variable SSTEST is monitored; this represents the maximum change in any dependent variable over a time. It should be noted that for this to be meaningful, the time step must be

significant. SSTEST can be made arbitrarily small by decreasing the time step. The most important criterion is the maximum residual. The residual of each equation is obtained by setting the time-derivative term to zero, and the placing all remaining terms on the right-hand side of the equation. The sum of all terms on the right-hand side of the equation defines the residual. Obviously, when the residual is zero the equations satisfy a steady state solution. Both the maximum residual throughout the domain for each equation and the average residual within the domain for each equation were monitored. These usually could be decreased by between two and four orders of magnitude during the run when steady solutions were sought.

However, even the presence of residuals requires interpretation. As prevously discussed, these could be indicative of flow unsteadiness. Also, relatively large residuals occur at the airfoil cusp trailing edge. In general, calculations for which steady solutions are sought are initiated from a very simple initial flow field. The initial flow field has constant pressure throughout and a velocity field identical to that at upstream infinity with a simple boundary layer correction. For steady flows converged solutions can be obtained within 70 time steps. The procedure suggested in the previous section may require 100 time steps.

In regard to run time, the current code is a general research type code which was created with flexibility in mind. Obviously, the price of flexibility is increased computer run time. In the present code the run time for a 141 × 39 grid is approximately 15 cpu secs per time step on a CYBER 203. The code used is not fully optimized for scalar operation and has no vectorization.

Further Comments on Run Protocol

The run protocol described above should allow calculations to be made for both steady and oscillating airfoils provided a proper grid is used. The grid must have adequate resolution. In general, the first point from the airfoil should be of the order of 10^{-5} chords from the surface and the streamwise spacing in the vicinity of the leading edge stagnation point should be 10^{-3} chords. It is recommended the CRDSRA be used to generate the grid. If computational problems occur for airfoils at modest

incidence two approaches may be fruitful. The dissipation factor, σ, termed AVISC, can be raised to 0.5 in starting the solution. However, it should eventually be lowered to at least a value of 0.10. The time step can be reduced to a value of 0.01. These changes may help in obtaining a solution. However, if an airfoil calculation at modest incidence does appear to be having difficulty the most probable cause of the problem would be the grid and this should be scrutinized for adequate resolution, smooth metric data and minimum angle between coordinate lines. This latter value in general should be limited to 45°.

For high incidence airfoils in which major separation is expected, the above run protocol should hold. However, if difficulty occurs an alternate strategy would be to start the calculation at modest incidence and gradually change the incidence through several runs. A second alternative strategy would converge the solution at modest incidence and then perform a time-dependent calculation for an oscillating airfoil through one-half of a period such that for $0 < \omega(t-t_o) < \Pi$

$$\alpha = \alpha_1 + \frac{(\alpha_2 - \alpha_1)}{2} [1 - \cos [\omega(t-t_o)]]$$

and for $\Pi < \omega(t-t_o)$

$$\alpha = \alpha_2$$

IMPLEMENTATION INSTRUCTIONS

Control Stream

The airfoil code has been developed on the NASA Langley Research Center computer and the current version is operational on the CYBER 205 VSOS2.1 operating system. A sample control stream for an initial start for this system follows:

```
ACCOUNTING CARD 1
ACCOUNTING CARD 2
ACCOUNTING CARD 3
GET(OLDPL=SRAIR)
UPDATE(P, C, L=A1234)
TOVPS(INPUT, C6UD=COMPILE, UN=******, PW=******, AC=******)
7/8/9
*C DAL.MGS3X3
7/8/9
ACCOUNTING CARD 4
ACCOUNTING CARD 5
ATTACH,COMPILE.
ATTACH(GE0012E)
COPY(GE0012E,TAPE21)
RETURN(GE0012E)
REQUEST(TAPE10/600,RT=W)
REQUEST(TAPE20/200,RT=W)
REQUEST(TAPE2/500,RT=W)
REQUEST(TAPE5/RT=W)
COPY(INPUT,TAPE5)
COPY(TAPE5,OUTPUT)
FORTRAN (L=OUTPUT/1000,I=COMPILE,B=BINARY/250,OPT=BL)
LOAD (BINARY,L=OUTPUT,CN=GO,1000,GRLPALL= )
GO.
SWITCH (TAPE10,REST10)
SWITCH (TAPE20,REST20)
TONOS(Z,UUUD=REST10,REST20,JCS="******","******","******")
7/8/9
DATA
6/7/8/9
```

The first three cards are accounting cards. The next instruction gets OLDPL and this is followed by update and transfer of files to the CYBER 205. The card *C DAL.MGS3X3 writes the required decks on the COMPILE file. After control is transferred to the CYBER 205 two accounting cards appear. A previously created geometry file, GE0012E, is attached, copied to TAPE21 and then returned. Restart files TAPE10 and TAPE20, are requested; and scratch file, TAPE2, and input file, TAPE5, are requested. The card input is copied to TAPE5 and then to OUTPUT. The program is loaded and executed. After program execution of the input parameter, IREPUN-1, TAPE10 and TAPE20 contain restart information which can be saved for a subsequent run.

In the case of a restart run the run stream is as follows:

```
ACCOUNTING CARD 1
ACCOUNTING CARD 2
ACCOUNTING CARD 3
GET(OLDPL=SRAIR)
UPDATE(P,C,L=A1234)
ATTACH(TAPE9=REST10)
ATTACH(TAPE19=REST20)
TOVPS(INPUT, C6UD=COMPILE, UUUD=TAPE9, TAPE19, UN=******,
    PW=******, AC=******)
7/8/9
*C DAL.MGS3X3
7/8/9
ACCOUNTING CARD 4
ACCOUNTING CARD 5
ATTACH(TAPE9)
ATTACH(TAPE19)
SWITCH(TAPE9,RT=W)
SWITCH(TAPE19,RT=W)
ATTACH(GE0012E)
  :
  :
  :
```

In this run stream files REST10 and REST20, which are previously created restart files, are ATTACHed and then passed to the CYBER 205. After control is passed to the CYBER 205, they must be ATTACHed again and the SWITCH command used to set the file type. Otherwise the run streams are identical.

The required input consists of a geometry file, TAPE21, restart files, TAPE9 and TAPE19, and card input data. The geometry and restart file input has been discussed previously and will not be discussed further. The present section considers the card input. Some card input data refers to arrays in which the indices may be IEQ, referring to equation, and IDIR, referring to coordinate direction. For purposes of this discussion

IEQ =	1	x-momentum equation
	3	z-momentum equation
	4	continuity equation
IDIR =	1	ξ-direction
	3	η-direction
LX,LZ		Grid points numbering for LX starts from point A' and proceeds to E and to D' with LX at A' being 1 and LX and D' being NX2. Grid point numbering for LZ starts at A' (or H) as 1 and proceeds to A (or I) where LZ = NZ2

NAMELIST READ 1

IREST =	0	Initial run
	1	Restart run
IREPUN =	0	Nothing written on TAPE10 and TAPE20
	1	Restart information written on TAPE10 and TAPE20

Note: In general set IREPUN=1 even if a restart file is not desired. Certain output information and plot files require the restart mechanism to be called. If no restart is desired, simply do not save the restart files.

NAMELIST READ2

ALPINF	Upstream incidence
IDBL	Last LX location prior to branch cut. This is the grid point number of the last point on the branch cut as one proceeds from A' (see Fig. 1) where LX = 1 to the airfoil trailing edge.
I4LE	Last LX location prior to frontal cap of outer loop. In general, the outer loop consists of two parallel straight lines joined by a frontal cap. I4LE is the last point on the straight line as one proceeds from A (LX = 1) to F.
NUMDX	Number of interior ξ-points; i.e., two less than the total number of ξ-points.
NUMDZ	Number of interior η-points; i.e. two less than the total number of η-points.

Note: ALPINF represents the negative of the incidence angle. For a horizontal NACA 0012 airfoil immersed in a flow of 6° with positive u & w components, $\alpha = -.105$.

NAMELIST READ3

CLENG	Reference length taken as airfoil chord.
DENSR	Reference density taken as free stream density.
TREF	Reference static temperature taken as free stream static temperature.

NAMELIST READ3

WREF	Reference velocity taken as free stream velocity.
VISCR	Reference viscosity taken as free stream viscosity.
AVISC(IDIR,IEQ)	Numerical dissipation factor. Recommend values for initial calcualtion AVISC = 15*0.1; for calculations near convergence try AVISC = 15*0.025.
NXBL(I),I=1,4	Turbulence model keys. Set NXBL(1) = NX2/2 - 10 NXBL(2) = NX2/2 + 10 NXBL(3) = IDBL + 1 NXBL(4) = NX2 - IDBL NXBL(1) and NXBL(2) should bracket the front stagnation point location. NXBL(3) and NXBL(4) should note the airfoil trailing edge.
NZDL	Turbulence model key. Set equal to NZ2-5. This limits the search for edge of boundary layer.
ALPAF	Incidence used for force calculations. Set equal to ALPINF.
XLTR1	Dimensionless location of suction side transition location. Default = -1.0.
XLTR2	Dimensionless location of pressure side transition location. Default = -1.0.
XLDTR	Dimensionless length of transition region. Default = -1.0.

Note: In regard to reference quantity CLENG, DENSR, TREF, WREF and VISCR, these are in general taken as km units. However, any consistent set which gives the desired Reynolds number and Mach number may be used. The Reynolds number, REY, and the Mach number, CMACH, are printed in NAMELIST READ3. If these are not as required, any of the above quantities may be modified to obtain the required Mach and Reynolds numbers. In regard to transition locations, if these are less than the location of the airfoil leading edge turbulence is assumed over all the airfoil. The leading edge is located at x/c = -0.25 to put the quarter chord location at x/c = 0.0.

NAMELIST READ4

DT	Initial time step.
DTMIN	Minimum time step.
DTMAX	Maximum time step.
IDUMP1	1 - Print initial flow field. 2 - Do not print initial flow field.
IVARPR	Print Flag IVARPR(1) - u velocity IVARPR(3) - w velocity IVARPR(4) - density IVARPR(25) - pressure IVARPR(27) - viscosity Set to 1 - print final array Set to 0 - no print Default - all set to 1

NAMELIST READ4

ITCALC	0 - March in physical time. 5 - Invoke time scaling.
NT	Number of time steps.
IPRINT	Print flag - set equal to NT.
PCNT1	Increase time step if maximum change in flow field on previous time step is less than PCNT1. Default PCNT1 = .04.
PCNT2	Decrease time step if maximum change in flow field on previous time step is greater than PCNT2. Default PCNT2 = .06.

Note: For cases in which steady solutions are sought set ITCALC = 5, DT = 1.0, DTMIN = 1.0, DTMAX = 10.0.

For cases in which transients are to be followed set ITCALC = 0, DT = .01, DTMIN = 0.01, DTMAX = .1.

In general, use ITCALC = 5 option from cold start and only use ITCALC = 0 after steady solution has nearly converged. Also depending upon the case low Mach number, time-accurate solutions may be difficult to obtain for M < 0.10.

DALPH TZR QOMEG	Parameters for oscillating airfoil
ITDGEO	

Note: The time-dependent motion is given by

$$\alpha = \alpha_o + \frac{\Delta\alpha}{2} \left[1 - \cos\omega\,(t-to)\right]$$

where

ALF1 = $-\alpha_o$, represents minimum incidence
DALDH = $-\Delta\alpha$
QOMEG = ω
TZR = t_o

As shown in Fig. 8, the oscillating airfoil motion is obtained by setting IDTGEO = 1.

NAMELIST BCSET

This NAMELIST allows resetting the default boundary conditions by specifying the array JEQBC(, ,). This array contains three indicators. IBC refers to the boundary where IBC=1,2,3,4.

IBC = 1	Boundary A'A
2	Boundary D'D
3	Boundary A'ED'
4	Boundary AFD

IV refers to the variable

IV =	1	u-velocity
	2	w-velocity
	3	Density

LN refers to the point number, the value of LX or LZ, on the boundary.

JEQBC =	1	Function condition
	2	First derivative condition
	3	Second derivative condition
	7	Static pressure (For density boundary condition only)
	14	Total pressure (For u-velocity boundary condition only)
	-2	Flow angle (For w-velocity boundary condition only).

It is recommended that the default conditions be used.

Output

The first item of output denotes whether the run is an initial or restart run. This is followed by dimensionless groupings DIM1-DIM10, DIM12, DIM14 and by dimensionless total temperature, enthalpy and pressure. This is followed by a listing of the difference operators. The next items are the namelists. This is followed by the boundary condition output which consists of thirteen columns. The first column is the grid point number. The next twelve are broken into four (4) groups of three columns each of which correspond to a boundary. The boundaries are lines A'A, D'D, A'ED' and AFD, respectively. Within each group of three, the first column represents the boundary condition for the x-momentum equation, the second column represents the boundary condition for the z-momentum equation, and the third column for the continuity equation. This represents the initial data.

The initial data is followed by print out on a time step by time step basis at each time step. If the flow is one in which the airfoil is oscillating ALPDEG is printed. This represents the negative change in incidence from the initial value in radians. If the airfoil is steady, this is not printed. In either case, the arrays for dimensionless pressure p/P_{ref} along the line LZ = 1, and the boundary layer distribution are printed. This is followed by three lines which gives the time step number, the time step increment, DT, and SSTEST. SSTEST represents the maximum change in any variable over the preceding time step. The location at which this occurs follows SSTEST on the same line with LX and LZ being grid point numbers. The next line gives the maximum change for variable (1) , variable (3) and variable (4) during the last step; these are u, w, and ρ, respectively.

This is followed by force coefficients FRCZ and FRCX here FRCZ and FRCX are interpreted as pressure force coefficients in the Z and X directions. This is followed by CMOM the moment about the quarter chord. The final item gives the average residual in the field, RESAVG, and the maximum residual, RESMAX, as well as its location. This is followed by average and maximum residual on an equation by equation basis.

After all time steps are completed, plot files are written and this noted and surface static pressure coefficients are given. This is followed by array output for u, w, ρ, p and μ_T as requested via IVARPR. In reading these arrays, LZ = 1 refers to line A'ED' of Fig. 1, and LZ = NZ2 refers to line AFD. LX = 1 refers to line A'A and LX = 2 refers to line D'D. The variables u, w, ρ and μ_T are normalized by reference quantities. The variable p is given by a pressure coefficient.

ENVIRONMENTAL CHARACTERISTICS

The program was developed on the CYBER 170 and CYBER 203. The current version is operational on the CYBER 205, and CYBER 203. A typical grid of 140 x 39 points requires 15 CPU seconds per time step.

REFERENCES

1. Shamroth, S.J. and Gibeling, H.J.: A Compressible Solution of the Navier-Stokes Equations for Turbulent Flow About an Airfoil. NASA CR-3183, 1979. (See also AIAA Paper 79-1543).

2. Shamroth, S.J. and Gibeling, H.J.: Analysis of Turbulent Flow About an Isolated Airfoil Using a Time-Dependent Navier-Stokes Procedure. Presented at AGARD Conference on Boundary Layer Effects on Unsteady Airfoils. AGARDograph 296, 1980.

3. Shamroth, S.J.: A Turbulent Flow Navier-Stokes Analysis for an Airfoil Oscillating in Pitch. IUTAM Symposium on Turbulent Shear Flows, Springer-Verlag, New York, 1981.

4. Shamroth, S.J.: Calculation of Steady and Oscillating Airfoil Flow Fields via the Navier-Stokes Equations. AIAA Paper 84-0525, 1984.

5. Shamroth, S.J.: A Navier-Stokes Calculation of the Airfoil Dynamic Stall Process, AFOSR/FJSRL/U. of Col. Workshop on Unsteady Separated Flows, 1983.

6. Shamroth, S.J.: Calculation of Steady and Unsteady Airfoil Flow Fields via the Navier-Stokes Equations. NASA CR-3899, 1985.

7. Weinberg, B.C., Yang, R.-J., McDonald, H. and Shamroth, S.J.: Calculation of Two- and Three-Dimensional Transonic Cascade Flow Fields. ASME Paper 85-GT-66, 1985.

8. Schlichting, H.: Boundary Layer Theory, McGraw-Hill, New York, 1960.

9. Briley, W.R. and McDonald, H.: Solution of the Multidimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method, J. Comp. Physics, Vol. 24, No. 4, August 1977, p. 372.

10. Lindemuth, L. and Killeen, J.: Alternating Direction Implicit Techniques for Two-Dimensional Magnetohydrodynamic Calculations, J. Comp. Physics, Vol.13, 1973, pp. 181-208.

11. Briley, W.R. and McDonald, H.: On the Structure and Use of Linearized Block Implicit Schemes, J. of Comp. Physics, Vol. 34, No. 1, Jan. 1980, pp. 54-73.

12. Douglas, J. and Gunn, J.E.: A General Formulation of Alternating Direction Methods, Numerische Math, Vol. 6, 1964, pp. 428-453.

13. Briley, W.R., Buggeln, R.C. and McDonald, H.: Computation of Laminar and Turbulent Flow in 90 Degree Square Duct and Pipe Bends Using the Navier-Stokes Equations. SRA Rpt. R82-920009-F, 1982.

14. Briley, W.R. and McDonald, H.: Computation of Three-Dimensional Horseshoe Vortex Flow Using the Navier-Stokes Equations. Seventh International Conference on Numerical Methods in Fluid Dynamics, 1980.

15. Liu, N.-S., Shamroth, S.J. and McDonald, H.: Numerical Solution of the Navier-Stokes Equations for Compressible Turbulent Two/Three Dimensional Flows in the Terminal Shock Region of an Inlet/Diffuser, AIAA Paper 83-1892, 1983.

16. McDonald, H., Shamroth, S.J. and Briley, W.R.: Transonic Flows with Viscous Effects. Transonic, Shock and Multidimensional Flows: Advances in Scientific Computing, Academic Press, New York, 1982.

17. Liu, N.-S., Shamroth, S.J. and McDonald, H.: Numerical Solutions of Navier-Stokes Equations for Compressible Turbulent Two/Three Dimensional Flows in the Terminal Shock Region of an Inlet/Diffuser, NASA CR-3723, 1983.

18. Beam, R.M. and Warming, R.F.: An Implicit Factored Scheme for the Compressible Navier-Stokes Equations. AIAA Journal, Vol. 16, No. 4, April 1978, pp. 393-402.

19. Beam, R.M. and Warming, R.F.: An Implicit Finite Difference Algorithm for Hyperbolic Systems in Conservation Law Form. Journal of Computational Physics, Vol. 22, 1976, pp. 87-110.

20. Beam, R.M. and Warming, R.F.: Alternating Direction Implicit Methods for Parabolic Equations with a Mixed Derivative. SIAM J. Sci. Stat. Comp., Vol. 1, 1980, pp. 131-157.

APPENDIX - SOLUTION PROCEDURE [13]

Background

The solution procedure employs a consistently-split linearized block implicit (LBI) algorithm which has been discussed in detail in [9, 11]. There are two important elements of this method:

(1) the use of a noniterative formal time linearization to produce a fully-coupled linear multidimensional scheme which is written in "block implicit" form; and

(2) solution of this linearized coupled scheme using a consistent "splitting" (ADI scheme) patterned after the Douglas-Gunn [12] treatment of scalar ADI schemes.

The method is thus referred to as a split linearized block implicit (LBI) scheme. The method has several attributes:

(1) the noniterative linearization is efficient;

(2) the fully-coupled linearized algorithm eliminates instabilities and/or extremely slow convergence rates often attributed to methods which employ ad hoc decoupling and linearization assumptions to identify nonlinear coefficients which are then treated by lag and update techniques;

(3) the splitting or ADI technique produces an efficient algorithm which is stable for large time steps and also provides a means for convergence acceleration for further efficiency in computing steady solutions;

(4) intermediate steps of the splitting are consistent with the governing equations, and this means that the "physical" boundary conditions can be used for the intermediate solutions. Other splittings which are inconsistent can have several difficulties in satisfying physical boundary conditions [11].

(5) the convergence rate and overall efficiency of the algorithm are much less sensitive to mesh refinement and redistribution than algorithms based on explicit schemes or which employ ad hoc decoupling and linearization assumptions. This is important for accuracy and for computing turbulent flows with viscous sublayer resolution; and

(6) the method is general and is specifically designed for the complex systems of equations which govern multiscale viscous flow in complicated geometries. This same algorithm was later considered by Beam and Warming [18], but the ADI splitting was derived by approximate factorization instead of the Douglas-Gunn procedure. They refer to the algorithm as a "delta form" approximate factorization scheme. This scheme replaced an earlier non-delta form scheme [19], which has inconsistent intermediate steps.

Spatial Differencing and Artificial Dissipation

The spatial differencing procedures used are a straightforward adaption of those used in [9] and elsewhere. Three-point central difference formulas are used for spatial derivatives, including the first-derivative convection and pressure gradient terms. This has an advantage over one-sided formulas in flow calculations subject to "two point" boundary conditions (virtually all viscous or subsonic flows), in that all boundary conditions enter the algorithm implicitly. In practical flow calculations, artificial dissipation is usually needed and is added to control high-frequency numerical oscillations which otherwise occur with the central-difference formula.

In the present investigation, artificial (anisotropic) dissipation terms of the form

$$\sum_j \frac{d_j}{h_j^2} \frac{\partial^2 u_k}{\partial x_j^2} \tag{1}$$

are added to the right-hand side of each (k-th) component of the momentum equation, where for each coordinate direction x_j, the artificial diffusivity d_j is positive and is chosen as the larger of zero and the local quantity μ_e (σ $Re_{\Delta x}$-1)/Re. Here, the local cell Reynolds number $Re_{\Delta x_j}$ for the j-th direction is defined by

$$Re_{\Delta x_j} = Re \left|\rho u_j\right| \Delta x_j / \mu_e \tag{2}$$

This treatment lowers the formal accuracy to 0 (Δx), but the functional form is such that accuracy in representing physical shear stresses in thin shear layers with small normal velocity is not seriously degraded. This

latter property follows from the anisotropic form of the dissipation and the combination of both small normal velocity and small grid spacing in thin shear layers.

Split LBI Algorithm

Linearization and Time Differencing

The system of governing equations to be solved consists of three/four equations: continuity and two/three components of momentum equation in three/four dependent variables: ρ, u, v, w. Using notation similar to that in [9], at a single grid point this system of equations can be written in the following form:

$$\partial H(\phi)/\partial t = D(\phi) + S(\phi) \tag{3}$$

where ϕ is the column-vector of dependent variables, H and S are column-vector algebraic functions of ϕ, and D is a column vector whose elements are the spatial differential operators which generate all spatial derivatives appearing in the governing equation associated with that element.

The solution procedure is based on the following two-level implicit time-difference approximations of (3):

$$(H^{n+1} - H^n)/\Delta t = \beta(D^{n+1} + S^{n+1}) \quad (1-\beta)\ (D^n + S^n) \tag{4}$$

where, for example, H^{n+1} denotes $H(\phi^{n+1})$ and $\Delta t = t^{n+1} - t^n$. The parameter β ($0.5 \leq \beta \leq 1$) permits a variable time-centering of the scheme, with a truncation error of order $[\Delta t^2, (\beta - 1/2)\ \Delta t]$.

A local time linearization (Taylor expansion about ϕ^n) of requisite formal accuracy is introduced, and this serves to define a linear differential operator L (cf. [9]) such that

$$D^{n+1} = D^n + L^n(\Phi^{n+1} - \Phi^n) + O(\Delta t^2) \tag{5}$$

Similarly,

$$H^{n+1} = H^n + (\partial H/\partial \phi)^n\ (\phi^{n+1} - \phi^n) + 0\ (\Delta t^2) \tag{6}$$

$$S^{n+1} = S^n + (\partial S/\partial \phi)^n\ (\phi^{n+1} - \phi^n) + 0\ (\Delta t^2) \tag{7}$$

Eqs. (5-7) are inserted into Eq. (4) to obtain the following system which is linear in ϕ^{n+1}

$$(A - \beta\Delta t\ L^n)\ (\phi^{n+1} - \phi^n) = \Delta t\ (D^n + S^n) \tag{8}$$

and which is termed a linearized block implilcit (LBI) scheme. Here, A denotes a matrix defined by

$$A \equiv (\partial H/\partial \phi)^n - \beta \Delta t \ (\partial S/\partial \phi)^n \tag{9}$$

Eq. (8) has 0 (Δt) accuracy unless $H \equiv \phi$, in which case the accuracy is the same as Eq. (4).

Special Treatment of Diffusive Terms

The time differencing of diffusive terms is modified to accomodate cross-derivative terms and also turbulent viscosity and artificial dissipation coefficients which depend on the solution variables. Although formal linearization of the convection and pressure gradient terms and the resulting implicit coupling of variables is critical to the stability and rapid convergence of the algorithm, this does not appear to be important for the turbulent viscosity and artificial dissipation coefficients. Since the relationship between μ_e and d_j and the mean flow variables is not conveniently linearized, these diffusive coefficients are evaluated explicitly at t^n during each time step. Notationally, this is equivalent to neglecting terms proportional to $\partial \ \mu_e/\partial \phi$ or $\partial d_j/\partial \phi$ in L^n, which are formally present in the Taylor expansion (5), but retaining all terms proportional to μ_e or d_j in both L^n and D^n.

It has been found through extensive experience that this has little if any effect on the performance of the algorithm. This treatment also has the added benefit that the turbulence model equations can be decoupled from the system of mean flow equations by an appropriate matrix partitioning [11] and solved separately in each step of the ADI solution procedure. This reduces the block size of the block tridiagonal systems which must be solved in each step and thus reduces the computational labor.

In addition, the viscous terms in the present formulation include a number of spatial cross-derivative terms. Although it is possible to treat cross-derivative terms implicitly within the ADI treatment which follows, it is not at all convenient to do so; and consequently, all cross-derivative terms are evaluated explicitly at t^n. For a scalar model equation representing combined convection and diffusion, it has been shown by Beam and Warming [20] that the explicit treatment of cross-derivative terms does not degrade the unconditional stability of the present algorithm. To preserve notational simplicity, it is understood that all cross-derivative terms appearing in Ln are neglected but are

retained in D^n. It is important to note that neglecting terms in L^n has no effect on steady solutions of Eq. (8), since $\phi^{n+1} - \phi^n \equiv 0$, and thus Eq. (8) reduces to the steady form of the equations: $D^n + S^n = 0$. Aside from stability considerations, the only effect of neglecting terms in L^n is to introduce an $0\ (\Delta t)$ truncation error.

Consistent Splitting of the LBI Scheme

To obtain an efficient algorithm, the linearized system (8) is split using ADI techniques. To obtain the split scheme, the multidimensional operator L is rewritten as the sum of three "one-dimensional" sub-operators L_i $(i = 1, 2, 3)$ each of which contains all terms having derivatives with respect to the i-th coordinate. The split form of Eq. (8) can be derived either as in [9, 11] by following the procedure described by Douglas and Gunn [12] in their generalization and unification of scalar ADI schemes, or using approximate factorization. For the present system of equations, the split algorithm is given by

$$(A - \beta\Delta t L_1^n)\ (\phi^* - \phi^n) = \Delta t\ (D^n + S^n) \tag{10a}$$

$$(A - \beta\Delta t L_2^n)\ (\phi^{**} - \phi^n) = A\ (\phi^* - \phi^n) \tag{10b}$$

$$(A - \beta\Delta t L_3^n)\ (\phi^{n+1} - \phi^n) = A\ (\phi^{**} - \phi^n) \tag{10c}$$

where ϕ^* and ϕ^{**} are consistent intermediate solutions. If spatial derivatives appearing in L_i and D are replaced by three-point difference formulas, as indicated previously, then each step in Eqs. (10a-c) can be solved by a block-tridiagonal elimination.

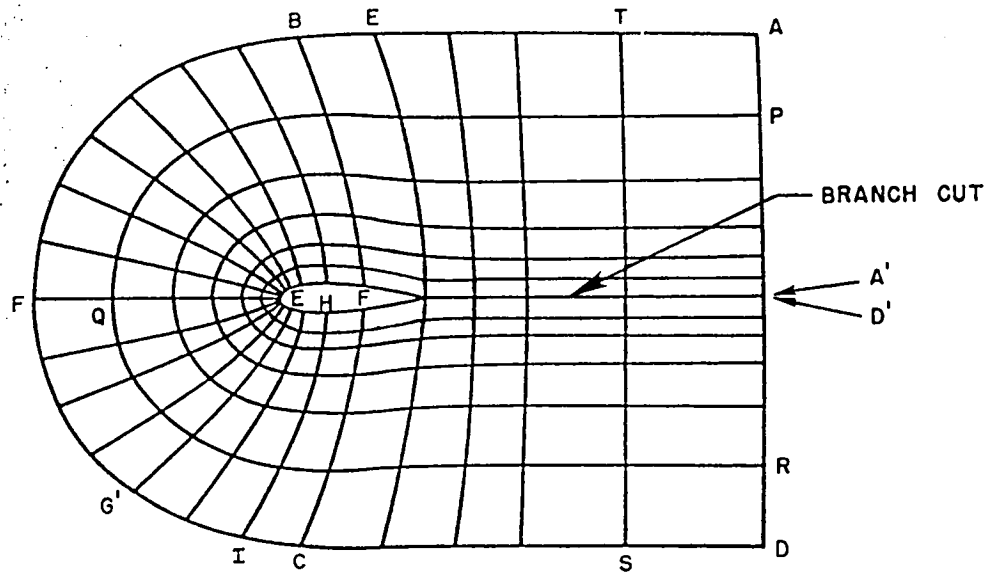
Combining Eqs. (10a-c) gives

$$(A - \beta\Delta t L_1^n)\ A^{-1}\ (A - \beta\Delta t L_2^n)\ A^{-1}\ (A - \beta\Delta t L_3^n)\ (\phi^{n+1} - \phi^n) = \Delta t\ (D^n + S^n) \tag{11}$$

which approximates the unsplit scheme (8) to $0\ (\Delta t^2)$. Since the intermediate steps are also consistent approximations for Eq. (8), physical boundary conditions can be used for ϕ^* and ϕ^{**} [9, 11]. Finally, since the L_i are homogeneous operators, it follows from Eqs. (10a-c) that steady solutions have the property that $\phi^{n+1} = \phi^* = \phi^{**} = \phi^n$ and satisfy

$$D^n + S^n = 0 \tag{12}$$

The steady solution thus depends only on the spatial difference approximations used for (12), and does not depend on the solution algorithm itself.


B
E
T
A
P
BRANCH CUT
A'
D'
F
Q
E
H
F
R
G'
I
C
S
D

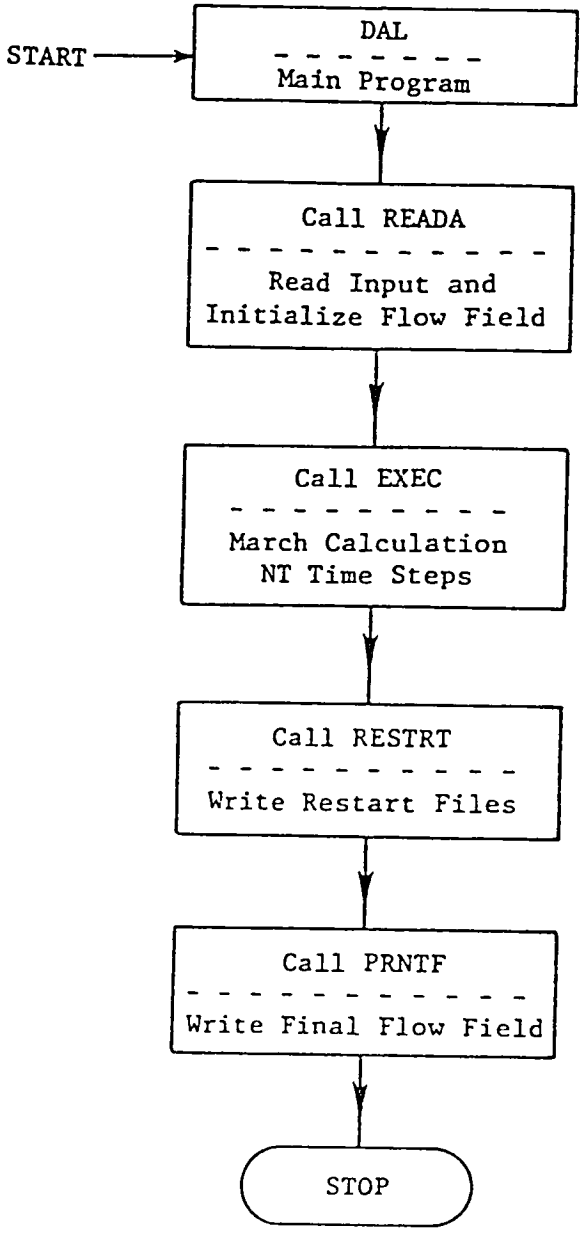


Fig. 2 - Overall Program Flow, PROGRAM DAL

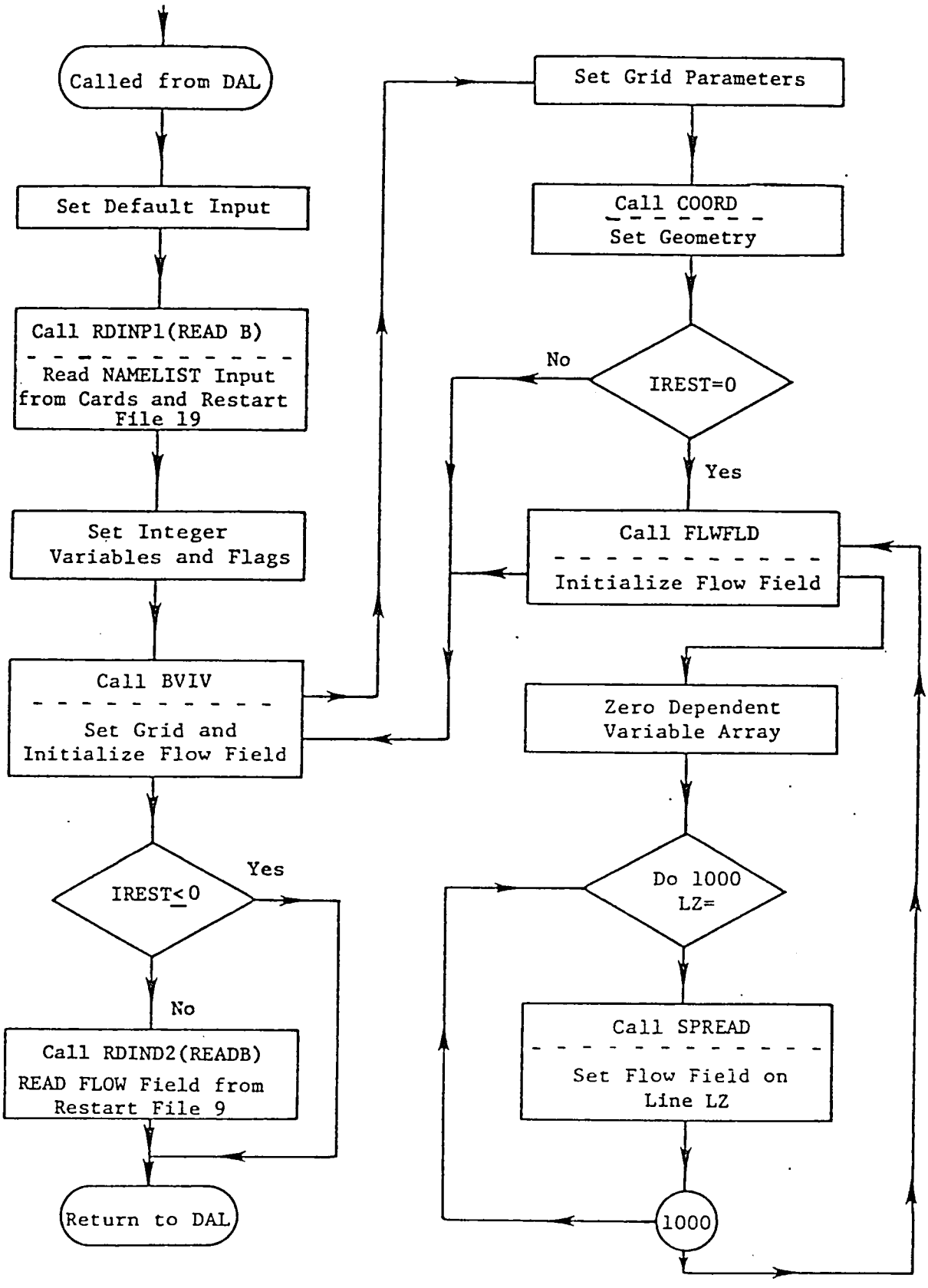


Fig. 3 - Program flow chart for SUBROUTINE READA

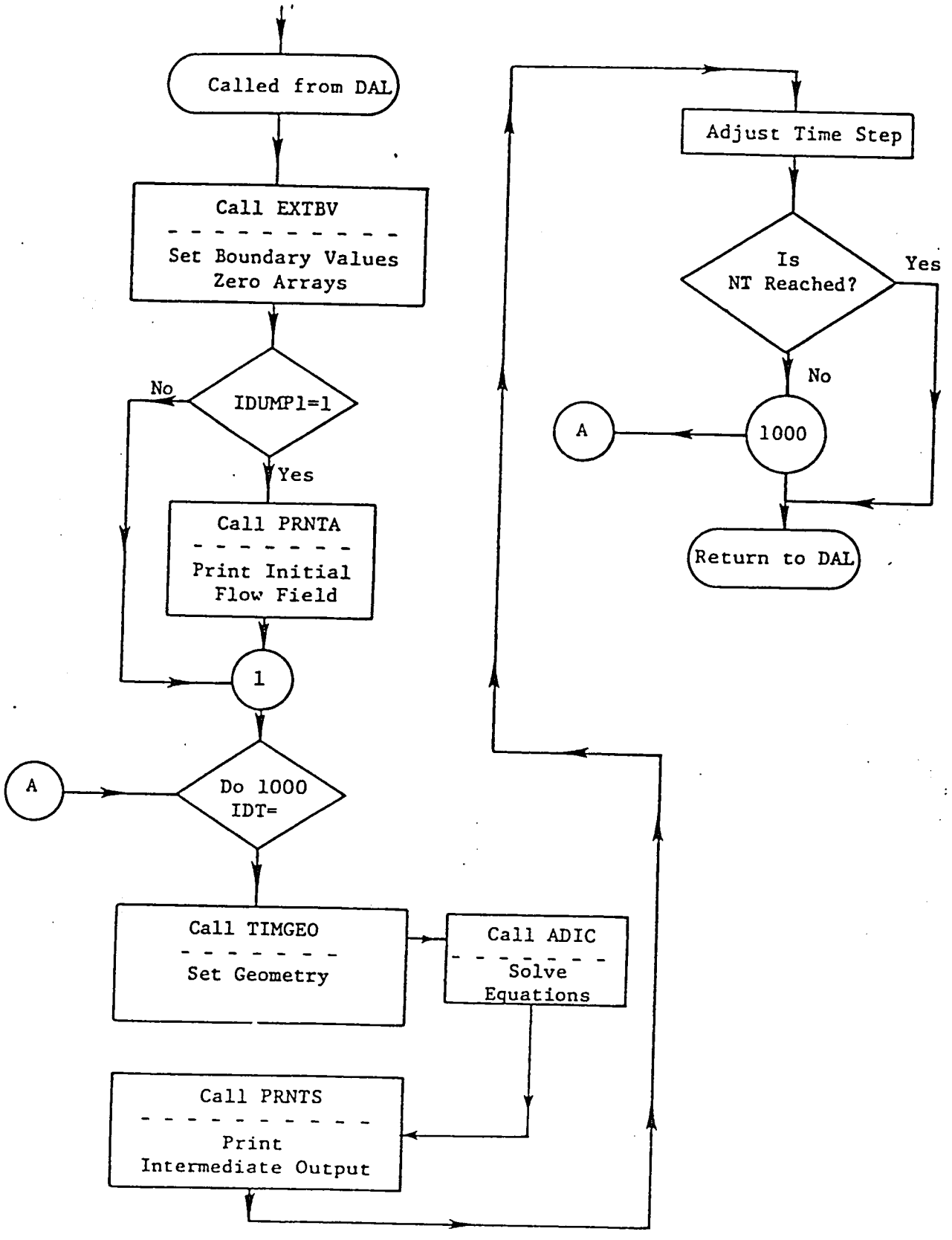


Fig. 4 - Program flow chart for SUBROUTINE EXEC

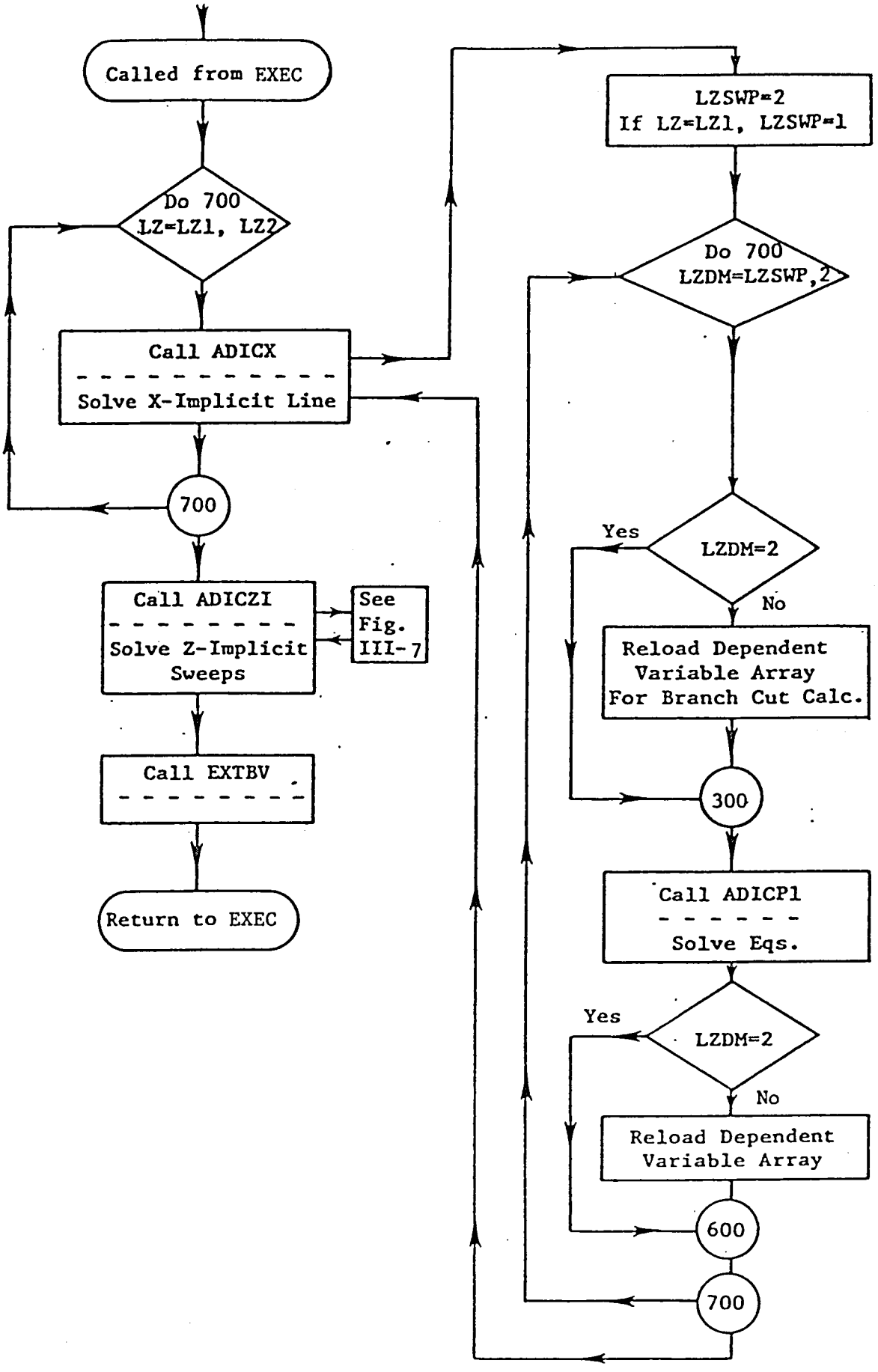


Fig. 5 - Program flow, SUBROUTINE ADIC

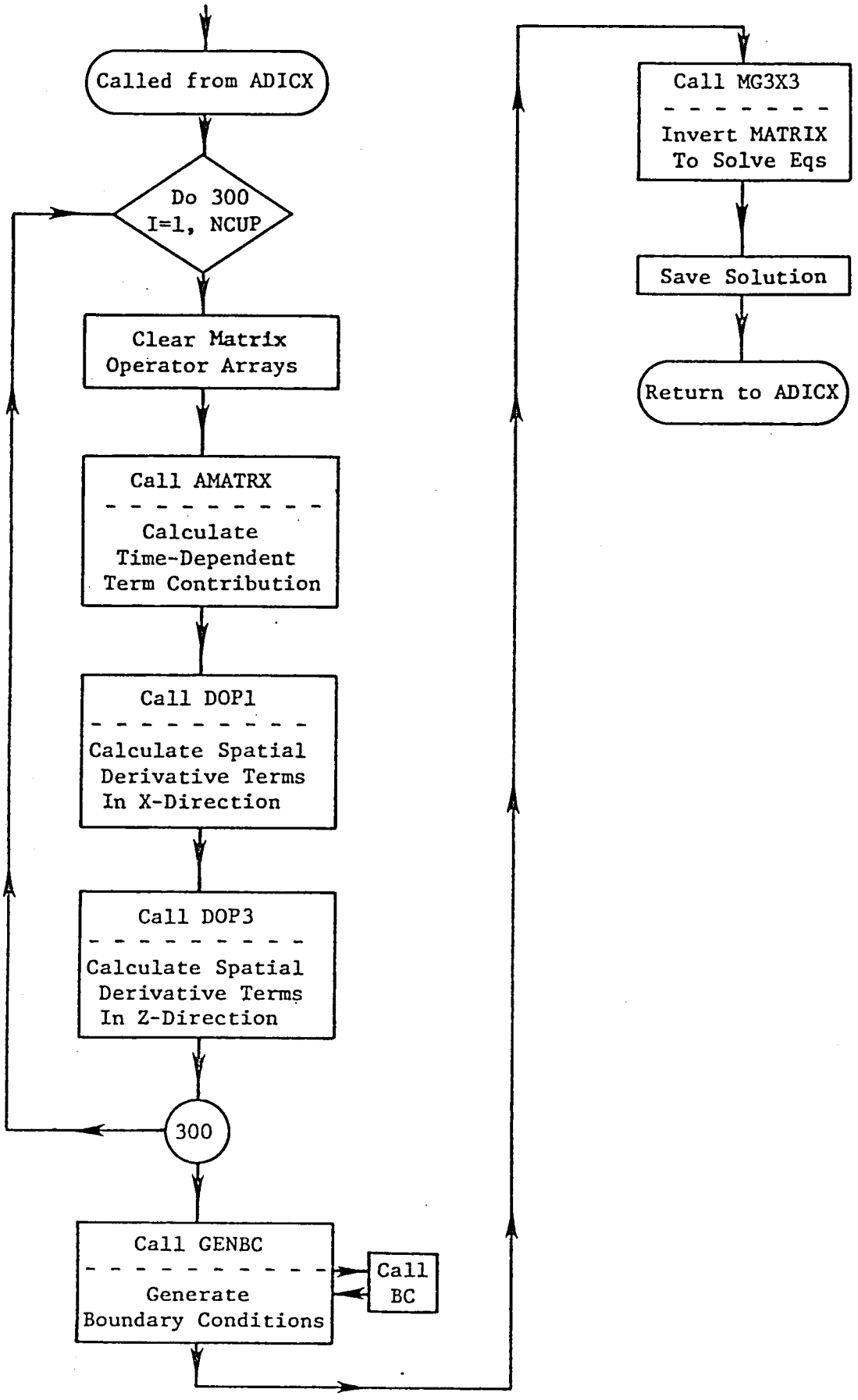


Fig. 6 - Program flow, SUBROUTINE ADICP1

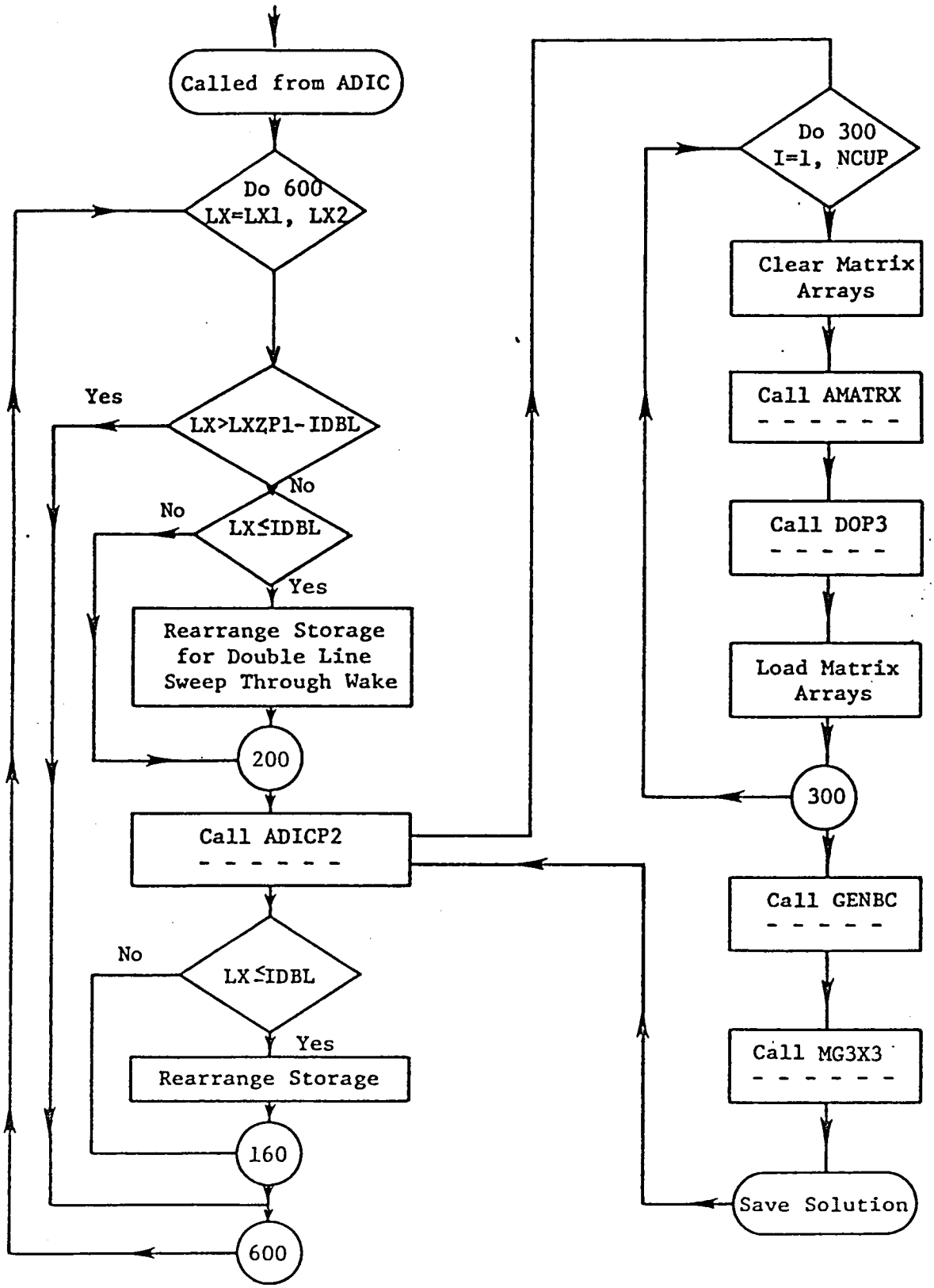


Fig. 7 - Program flow, SUBROUTINE ADICZI

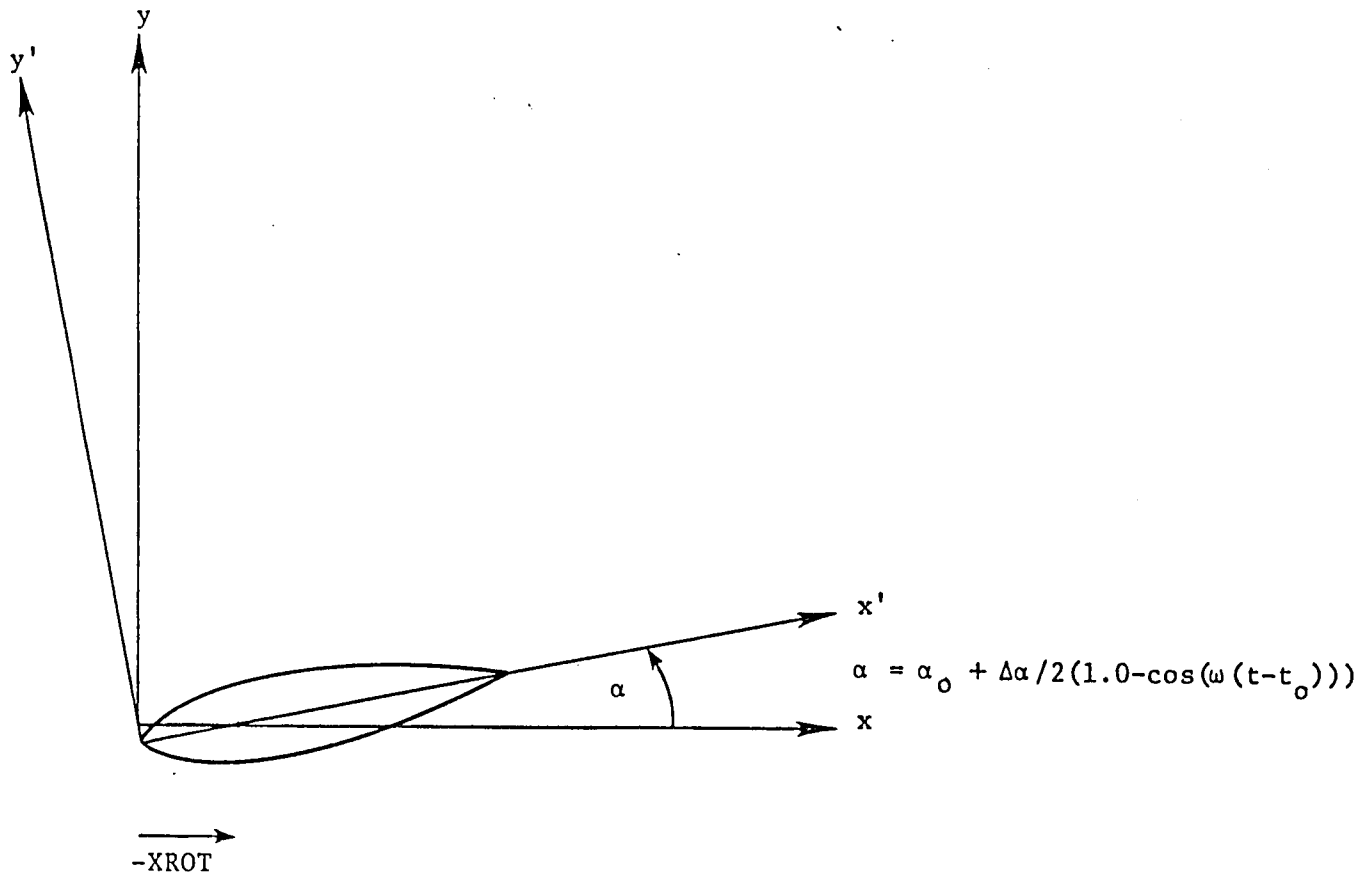


Fig. 8 - Pitching Motion Coordinate System

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-172585		

4. Title and Subtitle	5. Report Date
USER'S MANUAL FOR AIRFOIL FLOW FIELD COMPUTER CODE "SRAIR"	June 1985
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Stephen J. Shamroth	
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
Scientific Research Associates, Inc. P.O. Box 498 Glastonbury, CT 06033	NAS1-15214
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code
	505-33-43-09

15. Supplementary Notes

Langley Technical Monitor: James T. Howlett

16. Abstract

The present report describes a two-dimensional unsteady Navier-Stokes calculation procedure with specific application to the isolated airfoil problem. The procedure solves the full, ensemble-averaged Navier-Stokes equations with turbulence represented by a mixing length model. The equations are solved in a general nonorthogonal coordinate system which is obtained via an external source. Specific Cartesian locations of grid points are required as input for this code. The method of solution is based upon the Briley-McDonald LBI procedure. The manual discusses the analysis, flow of the program, control stream, input and output. The program as well as files containing sample input and output are being submitted to COSMIC.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Navier-Stokes Isolated Airfoil Unsteady Flow Dynamic Stall	Unclassified - Unlimited Subject Category 02

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	59	A04

For sale by the National Technical Information Service, Springfield, Virginia 22161