Proteus Two-Dimensional Navier-Stokes Computer Code—Version 2.0

Volume 3—Programmer's Reference

Charles E. Towne, John R. Schwab, and Trong T. Bui
Lewis Research Center
Cleveland, Ohio

October 1993
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUMMARY</td>
<td>3</td>
</tr>
<tr>
<td>1.0 INTRODUCTION</td>
<td>5</td>
</tr>
<tr>
<td>2.0 PROGRAM STRUCTURE</td>
<td>7</td>
</tr>
<tr>
<td>2.1 FLOW CHART</td>
<td>7</td>
</tr>
<tr>
<td>2.2 SUBPROGRAM CALLING TREE</td>
<td>10</td>
</tr>
<tr>
<td>2.3 PROGRAMMING CONVENTIONS AND NOTES</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 Computer &amp; Language</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2 Fortran Variables</td>
<td>15</td>
</tr>
<tr>
<td>3.0 COMMON BLOCKS</td>
<td>19</td>
</tr>
<tr>
<td>3.1 COMMON BLOCK SUMMARY</td>
<td>19</td>
</tr>
<tr>
<td>3.2 COMMON VARIABLES LISTED ALPHABETICALLY</td>
<td>19</td>
</tr>
<tr>
<td>3.3 COMMON VARIABLES LISTED SYMBOLICALLY</td>
<td>38</td>
</tr>
<tr>
<td>4.0 PROTEUS SUBPROGRAMS</td>
<td>49</td>
</tr>
<tr>
<td>4.1 SUBPROGRAM SUMMARY</td>
<td>49</td>
</tr>
<tr>
<td>4.2 SUBPROGRAM DETAILS</td>
<td>51</td>
</tr>
<tr>
<td>Subroutine ADI</td>
<td>53</td>
</tr>
<tr>
<td>Subroutine AVISC1</td>
<td>54</td>
</tr>
<tr>
<td>Subroutine AVISC2</td>
<td>57</td>
</tr>
<tr>
<td>Subroutine BCDENS</td>
<td>60</td>
</tr>
<tr>
<td>Subroutine BCELIM</td>
<td>63</td>
</tr>
<tr>
<td>Subroutine BCF</td>
<td>64</td>
</tr>
<tr>
<td>Subroutine BCFLIN</td>
<td>69</td>
</tr>
<tr>
<td>Subroutine BCGEN</td>
<td>71</td>
</tr>
<tr>
<td>Subroutine BCGRAD</td>
<td>73</td>
</tr>
<tr>
<td>Subroutine BCMET</td>
<td>74</td>
</tr>
<tr>
<td>Subroutine BCRES</td>
<td>75</td>
</tr>
<tr>
<td>Subroutine BCQ</td>
<td>82</td>
</tr>
<tr>
<td>Subroutine BCSET</td>
<td>85</td>
</tr>
<tr>
<td>Subroutine BCTEMP</td>
<td>87</td>
</tr>
<tr>
<td>Subroutine CUCVEL</td>
<td>93</td>
</tr>
<tr>
<td>Subroutine CVDIR</td>
<td>97</td>
</tr>
<tr>
<td>Subroutine CVVEL</td>
<td>106</td>
</tr>
<tr>
<td>Subroutine CVWVL</td>
<td>110</td>
</tr>
<tr>
<td>Subroutine BLIN1</td>
<td>114</td>
</tr>
<tr>
<td>Subroutine BLIN2</td>
<td>116</td>
</tr>
<tr>
<td>Subroutine BLKOUT</td>
<td>117</td>
</tr>
<tr>
<td>Subroutine BLK2</td>
<td>118</td>
</tr>
<tr>
<td>Subroutine BLK2P</td>
<td>120</td>
</tr>
<tr>
<td>Subroutine BLK3</td>
<td>122</td>
</tr>
<tr>
<td>Subroutine BLK3P</td>
<td>124</td>
</tr>
<tr>
<td>Subroutine BLK4</td>
<td>126</td>
</tr>
<tr>
<td>Subroutine BLK4P</td>
<td>127</td>
</tr>
<tr>
<td>Subroutine BLK5</td>
<td>128</td>
</tr>
<tr>
<td>Subroutine BLK5P</td>
<td>129</td>
</tr>
<tr>
<td>BLOCK DATA</td>
<td>130</td>
</tr>
<tr>
<td>Subroutine BLOUT1</td>
<td>132</td>
</tr>
<tr>
<td>Subroutine BLOUT2</td>
<td>135</td>
</tr>
</tbody>
</table>
A computer code called Proteus has been developed to solve the two-dimensional planar or axisymmetric, Reynolds-averaged, unsteady compressible Navier-Stokes equations in strong conservation law form. The objective in this effort has been to develop a code for aerospace propulsion applications that is easy to use and easy to modify. Code readability, modularity, and documentation have been emphasized.

The governing equations are written in Cartesian coordinates and transformed into generalized nonorthogonal body-fitted coordinates. They are solved by marching in time using a fully-coupled alternating-direction-implicit solution procedure with generalized first- or second-order time differencing. The boundary conditions are also treated implicitly, and may be steady or unsteady. Spatially periodic boundary conditions are also available. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. Turbulence is modeled using either an algebraic or two-equation eddy viscosity model.

The program contains many operating options. The governing equations may be solved for two-dimensional planar flow, or axisymmetric flow with or without swirl. The thin-layer or Euler equations may be solved as subsets of the Navier-Stokes equations. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used to damp pre- and post-shock oscillations in supersonic flow and to minimize odd-even decoupling caused by central spatial differencing of the convective terms in high Reynolds number flow. Several time step options are available for convergence acceleration, including a locally variable time step and global time step cycling. Simple Cartesian or polar grids may be generated internally by the program. More complex geometries require an externally generated computational coordinate system.

The documentation is divided into three volumes. Volume 1 is the Analysis Description, and presents the equations and solution procedure used in Proteus. It describes in detail the governing equations, the turbulence model, the linearization of the equations and boundary conditions, the time and space differencing formulas, the ADI solution procedure, and the artificial viscosity models. Volume 2 is the User's Guide, and contains information needed to run the program. It describes the program's general features, the input and output, the procedure for setting up initial conditions, the computer resource requirements, the diagnostic messages that may be generated, the job control language used to run the program, and several test cases. Volume 3, the current volume, is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

1.0 INTRODUCTION

Much of the effort in applied computational fluid dynamics consists of modifying an existing program for whatever geometries and flow regimes are of current interest to the researcher. Unfortunately, nearly all of the available non-proprietary programs were started as research projects with the emphasis on demonstrating the numerical algorithm rather than ease of use or ease of modification. The developers usually intend to clean up and formally document the program, but the immediate need to extend it to new geometries and flow regimes takes precedence.

The result is often a haphazard collection of poorly written code without any consistent structure. An extensively modified program may not even perform as expected under certain combinations of operating options. Each new user must invest considerable time and effort in attempting to understand the underlying structure of the program if intending to do anything more than run standard test cases with it. The user’s subsequent modifications further obscure the program structure and therefore make it even more difficult for others to understand.

The Proteus two-dimensional Navier-Stokes computer program is a user-oriented and easily-modifiable flow analysis program for aerospace propulsion applications. Readability, modularity, and documentation were primary objectives during its development. The entire program was specified, designed, and implemented in a controlled, systematic manner. Strict programming standards were enforced by immediate peer review of code modules; Kernighan and Plauger (1978) provided many useful ideas about consistent programming style. Every subroutine contains an extensive comment section describing the purpose, input variables, output variables, and calling sequence of the subroutine. With just three clearly-defined exceptions, the entire program is written in ANSI standard Fortran 77 to enhance portability. A master version of the program is maintained and periodically updated with corrections, as well as extensions of general interest (e.g., turbulence models.)

The Proteus program solves the unsteady, compressible, Reynolds-averaged Navier-Stokes equations in strong conservation law form. The governing equations are written in Cartesian coordinates and transformed into generalized nonorthogonal body-fitted coordinates. They are solved by marching in time using a fully-coupled alternating-direction-implicit (ADI) scheme with generalized time and space differencing (Briley and McDonald, 1977; Beam and Warming, 1978). Turbulence is modeled using either the Baldwin and Lomax (1978) algebraic eddy-viscosity model or the Chien (1982) two-equation model. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. The boundary conditions are treated implicitly, and may be steady or unsteady. Spatially periodic boundary conditions are also available.

The program contains many operating options. The governing equations may be solved for two-dimensional planar flow, or axisymmetric flow with or without swirl. The thin-layer or Euler equations may be solved as subsets of the Navier-Stokes equations. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used to damp pre- and post-shock oscillations in supersonic flow and to minimize odd-even decoupling caused by central spatial differencing of the convective terms in high Reynolds number flow. Several time step options are available for convergence acceleration, including a locally variable time step and global time step cycling. Simple grids may be generated internally by the program; more complex geometries require external grid generation, such as that developed by Chen and Schwab (1988).

The documentation is divided into three volumes. Volume 1 is the Analysis Description, and presents the equations and solution procedure used in Proteus. It describes in detail the governing equations, the turbulence model, the linearization of the equations and boundary conditions, the time and space differencing formulas, the ADI solution procedure, and the artificial viscosity models. Volume 2 is the User’s Guide, and contains information needed to run the program. It describes the program’s general features, the input and output, the procedure for setting up initial conditions, the computer resource requirements,
the diagnostic messages that may be generated, the job control language used to run the program, and several test cases. Volume 3, the current volume, is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.


The authors would like to acknowledge the significant contributions made by their co-workers. Tom Benson provided part of the original impetus for the development of Proteus, and did the original coding of the block tri-diagonal inversion routines. Simon Chen did the original coding of the Baldwin-Lomax turbulence model, and consulted in the implementation of the nonlinear coefficient artificial viscosity model. William Kunik developed the original code for computing the metrics of the generalized nonorthogonal grid transformation. Frank Molls has created separate diagonalized and patched-grid versions of the code. Ambady Suresh did the original coding for the second-order time differencing and for the nonlinear coefficient artificial viscosity model. These people, along with Dick Cavicchi, Julie Conley, Jason Solbeck, and Pat Zeman, have also run many debugging and verification cases.
2.0 PROGRAM STRUCTURE

2.1 FLOW CHART

In this section, a flow chart is presented showing the overall sequence of tasks performed by the two-dimensional *Proteus* computer code. Depending on the various options used in a particular run, of course, some of the elements in the chart may be skipped.

![Flow chart for the 2-D Proteus computer code.](image)

Figure 2.1 - Flow chart for the 2-D Proteus computer code.
Figure 2.1 - Continued.
Figure 2.1 - Concluded.
### 2.2 SUBPROGRAM CALLING TREE

In this section, the calling sequence for the various subprograms in the *Proteus* 2-D code is shown using a tree structure. The subheadings correspond to the elements of the flow chart shown in the previous section. The main program, listed in the first column, calls the subprograms in the second column, which in turn call those in the third column, etc.\(^1\) For any given case, of course, some of these routines will not be used. The subprograms needed for a particular case will depend on the combination of input parameters being used. The individual subprograms are described in detail in Section 4.0.

<table>
<thead>
<tr>
<th>INITIALIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read and print input.</td>
</tr>
<tr>
<td>MAIN</td>
</tr>
<tr>
<td>Get grid and metric parameters.</td>
</tr>
<tr>
<td>MAIN</td>
</tr>
<tr>
<td>Get initial flow field.</td>
</tr>
<tr>
<td>MAIN</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Set point-by-point boundary condition values.

| MAIN | BCSET |

---

\(^1\) Throughout this Programmer’s Reference, elements of the Fortran language, such as input variables and subprogram names, are printed in the text using uppercase letters. However, in most implementations, Fortran is case-insensitive. The *Proteus* source code itself is written in lowercase.
**Initialize plot files and print initial or restart flow field.**

<table>
<thead>
<tr>
<th>MAIN</th>
<th>PLOT OUTPUT</th>
<th>VORTEX PRTOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SET UP FOR TIME STEP**

Compute time step size.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>TIMSTP</th>
<th>ISAMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset boundary conditions if time-dependent.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>TBC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FILL BLOCK COEFFICIENT MATRIX**

Add extra data line at \( N + 1 \) if spatially periodic in sweep direction.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compute coefficients of governing equations.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>EQSTAT</th>
<th>COEFC</th>
<th>COEFX</th>
<th>COEFE</th>
<th>COEFZ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add boundary conditions.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>EQSTAT</th>
<th>BCGEN</th>
<th>BCQ</th>
<th>BCGRAD</th>
<th>BCGMET</th>
<th>BCGRAD</th>
<th>BCMET</th>
<th>BCF</th>
<th>BCFLIN</th>
<th>BCMET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Proteus 2-D Programmer's Reference**

**2.2 Subprogram Calling Tree**

11
Compute residuals without artificial viscosity terms (sweep 1 only.)

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>RESID</th>
<th>SNRM2</th>
<th>ISAMAX</th>
<th>SASUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add artificial viscosity.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>AVISC1</th>
<th>AVISC2</th>
<th>BLKOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compute residuals with artificial viscosity terms (sweep 1 only.)

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>RESID</th>
<th>SNRM2</th>
<th>ISAMAX</th>
<th>SASUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SOLVE DIFFERENCE EQUATIONS**

Perform matrix inversion.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>ADI</th>
<th>BLKOUT</th>
<th>FILTER</th>
<th>ISAMAX</th>
<th>ISRCHEQ</th>
<th>BLKOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLK3P</td>
<td>FILTER</td>
<td>ISAMAX</td>
<td>ISRCHEQ</td>
<td>BLKOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLK3</td>
<td>FILTER</td>
<td>ISAMAX</td>
<td>ISRCHEQ</td>
<td>BLKOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLK4P</td>
<td>FILTER</td>
<td>ISAMAX</td>
<td>ISRCHEQ</td>
<td>BLKOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLK4</td>
<td>FILTER</td>
<td>ISAMAX</td>
<td>ISRCHEQ</td>
<td>BLKOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLK5P</td>
<td>FILTER</td>
<td>ISAMAX</td>
<td>ISRCHEQ</td>
<td>BLKOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLK5</td>
<td>FILTER</td>
<td>ISAMAX</td>
<td>ISRCHEQ</td>
<td>BLKOUT</td>
</tr>
</tbody>
</table>

|      |      | UPDATE |        |        |        |         |        |
|      |      |        |        |        |        |         |        |
Update boundary values from first sweep.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EXEC</th>
<th>BVUP</th>
<th>EQSTAT</th>
<th>BCQ</th>
<th>BCMET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCGEN</td>
<td></td>
<td>BCGRAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCUVEL</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCVVEL</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCWVEL</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCPRES</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCTEMP</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCDENS</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCVDIR</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCF</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BFLIN</td>
<td>BCMET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BLKOUT</td>
<td>BCMET</td>
</tr>
</tbody>
</table>

| Update auxiliary variables. |

<table>
<thead>
<tr>
<th>MAIN</th>
<th>EQSTAT</th>
<th>FTEMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Update turbulence parameters. |

<table>
<thead>
<tr>
<th>MAIN</th>
<th>TURBBL</th>
<th>VORTEX</th>
<th>ISAMAX</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BLOUT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLIN1</td>
<td>ISAMIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLOUT2</td>
<td>ISRCHEQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TURBCH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLIN2</td>
<td>ISAMAX</td>
<td>COEFS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YPLUSN</td>
<td>ISAMIN</td>
<td>BLK2P</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRODCT</td>
<td>ISRCHEQ</td>
<td>BLK2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXECT</td>
<td></td>
<td>COEFS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BLK2P</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BLK2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UPDTKE</td>
</tr>
</tbody>
</table>

| Check for convergence, and get CPU time remaining. |

<table>
<thead>
<tr>
<th>MAIN</th>
<th>CONV</th>
<th>TREMAIN</th>
<th>ISAMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### GENERATE OUTPUT

Print flow field output.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>OUTPUT</th>
<th>VORTEX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PRTOUT</td>
<td></td>
</tr>
</tbody>
</table>

Write plot and restart files.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>PLOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REST</td>
</tr>
</tbody>
</table>

Print convergence history.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>PRTHST</th>
</tr>
</thead>
</table>

### 2.3 PROGRAMMING CONVENTIONS AND NOTES

#### 2.3.1 Computer & Language

At NASA Lewis Research Center, Proteus is normally run on a Cray X-MP or Y-MP computer. With just three known exceptions, it is written entirely in ANSI standard Fortran 77 as described in the *CF77 Compiling System, Volume 1: Fortran Reference Manual* (Cray Research, Inc., 1990). The first exception is the use of namelist input. With namelist input, it’s relatively easy to create and/or modify input files, to read the resulting files, and to program default values. Since most Fortran compilers allow namelist input, its use is not considered a serious problem. The second exception is the use of *CALL statements to include *COMDECK’s, which contain the labeled common blocks, in most of the subprograms. This is a Cray UPDATE feature, and therefore the source code must be processed by UPDATE to create a file that can be compiled. UPDATE is described in the *UPDATE Reference Manual* (Cray Research, Inc., 1988). Since using the *CALL statements results in cleaner, more readable code, and since many computer systems have an analogous feature, the *CALL statements were left in the program. The third exception is the use of lowercase alphabetic characters in the Fortran source code. This makes the code easier to read, and is a common extension to Fortran 77.

Several library subroutines are called by Proteus. SGEFA and SGESL are Cray versions of LINPACK routines. SASUM and SNRM2 are Cray Basic Linear Algebra Subprograms (BLAS). ISAMAX, ISAMIN, and ISRCHEQ are Cray search routines. TREMAIN is a Cray Fortran library routine. All of these routines are described in detail in Section 4.0. In addition, SGEFA and SGESL are described in *Volume 3: UNICOS Math and Scientific Library Reference Manual* (Cray Research, Inc., 1989b) and by Dongarra, Moler, Bunch, and Stewart (1979); SASUM, SNRM2, ISAMAX, ISAMIN, and ISRCHEQ are described in *Volume 3: UNICOS Math and Scientific Library Reference Manual* (Cray Research, Inc., 1989b); and TREMAIN is described in *Volume 1: UNICOS Fortran Library Reference Manual* (Cray Research, Inc., 1989a).

The Proteus code is highly vectorized for optimal performance on the Cray. The coefficient generation is vectorized in the ADI sweep direction. Since the coefficient matrix is block tridiagonal, the equations are solved using the Thomas algorithm. This algorithm is recursive, and therefore cannot be vectorized in the sweep direction. However, by storing the coefficients and source terms in both coordinate directions, the algorithm can be vectorized in the non-sweep direction. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.

---

2 See the example in Section 8.1 of Volume 2.
2.3.2 Fortran Variables

Variable Names

In developing Proteus, code readability has been emphasized. We have therefore attempted to choose Fortran variable names that are meaningful. In general, they either match the notation used in the analysis description in Volume 1, or are in some way descriptive of the parameter being represented. For example, RHO, U, V, W, and ET are the Fortran variables representing the density \( \rho \), the velocities \( u, v, \) and \( w \), and the total energy per unit volume \( E_T \).

Real and Integer Variables

In general, the type (real or integer) of the Fortran variables follows standard Fortran convention (i.e., those starting with I, J, K, L, M, or N are integer, and the remainder are real.) There are, however, several variables that would normally be integer but are explicitly declared to be real. These are noted in the input description in Section 3.0 of Volume 2, and in the description of common block variables in Section 3.0 of this volume.

Array Dimensions

Most Fortran arrays are dimensioned using dimensioning parameters. These parameters are set in COMDECK PARAMS1. This allows the code to be re-dimensioned simply by changing the appropriate parameters, and then recompiling the entire program. The dimensioning parameters are described in Section 6.2 of Volume 2.

Initialization

All of the input Fortran variables, plus some additional variables, are initialized in BLOCK DATA. Most of the input variables are initialized to their default values directly, but some are initialized to values that trigger the setting of default values in subroutine INPUT. On the Cray X-MP and Y-MP at NASA Lewis, all uninitialized variables have the value zero. There are no known instances in the Proteus code, however, in which a variable is used before it is assigned a value.

Nondimensionalization

In general, Fortran variables representing physical quantities, such as RHO, U, etc., are nondimensional. Two types of nondimensionalizing factors are used - reference conditions and normalizing conditions. The factors used to nondimensionalize the governing equations in Section 2.0 of Volume 1 are called normalizing conditions. These normalizing conditions are defined by six basic reference conditions, for length, velocity, temperature, density, viscosity, and thermal conductivity, which are specified by the user. The normalizing conditions used in Proteus are listed in Table 3-1 of Volume 2.

Note that for some variables, like pressure, the normalizing condition is dictated by the form of the governing equations once the six basic reference conditions are chosen. Unfortunately, some of these may not be physically meaningful or convenient for use in setting up input conditions. Therefore, some additional reference conditions are defined from the six user-supplied ones. The reference conditions are listed in Table 3-2 of Volume 2.

Throughout most of the Proteus code, physical variables are nondimensionalized by the normalizing conditions. For input and output, however, variables are nondimensionalized by the reference conditions because they are usually more physically meaningful for the user. The Fortran variables representing the reference conditions themselves are, of course, dimensional.

One-Dimensional Addressing of Two-Dimensional Arrays

In the solution algorithm used in Proteus, there are several instances in which the same steps must be followed in both ADI sweep directions. An example is the computation, in the COEFC, COEFX, COEFY, COEFZ, and COEFE routines, of the submatrices in the block tridiagonal coefficient matrix. These computations involve various flow variables, such as RHO, U, etc., and metric quantities, such as
XIX, XIY, etc. These are stored as two-dimensional arrays, with the two subscripts representing, in order, the indices in the computational $\xi$ and $\eta$ directions. For the first ADI sweep, values at various $\xi$ indices are needed at a fixed $\eta$ index. For the second ADI sweep, the reverse is true. In order to use the same coding for both sweeps, a scheme for one-dimensional addressing of a two-dimensional array has been used.\(^3\)

In Fortran, multi-dimensional arrays are actually stored in memory as a one-dimensional sequence of values, with the first subscript incremented over its range first, then the second subscript, etc. We take advantage of this in Proteus. As a first step, the two-dimensional array is equivalenced to a one-dimensional array of the same total length. The one-dimensional array name is derived from the two-dimensional array name by adding a "1". Thus, letting $F$ represent a typical two-dimensional array,

\[
\text{dimension } f(nlp,n2p),f1(ntotp)
\]

\[
\text{equivalence } (f(1,1),f1(1))
\]

where $N1P$ and $N2P$ are dimensioning parameters specifying the dimension size in the $\xi$ and $\eta$ directions, and $NTOTP$ is a dimensioning parameter equal to $N1P \times N2P$. Next, we define a "step factor", which depends on the ADI sweep, and a "base index" which depends on the index in the non-sweep direction. For the first ADI sweep,

\[
\text{istep } = 1
\]

\[
\text{do 1000 } i2 = 2,npt2-1
\]

\[
iv = i2
\]

\[
\text{ibase } = 1 + (i2-1) \times nlp
\]

\[
\ldots
\]

\[
\text{1000 continue}
\]

And for the second ADI sweep,

\[
\text{istep } = nlp
\]

\[
\text{do 2000 } il = 2,nptl-1
\]

\[
iv = il
\]

\[
\text{ibase } = il
\]

\[
\ldots
\]

\[
\text{2000 continue}
\]

In both of the above examples, the loop is in the non-sweep direction and $IV$ therefore represents the index in the non-sweep direction. Nested inside this loop is a loop in the sweep direction. In this inner loop, we can compute the equivalent one-dimensional address for a location in a two-dimensional array from the step factor, the base index, and the index in the sweep direction. Thus, for either ADI sweep, the inner loop looks like

\[
\text{do 100 } i = 2,npts-1
\]

\[
\text{iiml } = \text{ibase } + \text{istep} \times (i-2)
\]

\[
\text{ii } = \text{ibase } + \text{istep} \times (i-1)
\]

\[
\text{iipl } = \text{ibase } + \text{istep} \times i
\]

\[
\ldots
\]

\[
\text{100 continue}
\]

where $I$ represents the index in the sweep direction. With this coding, for the first sweep.

\[^3\] An alternative would be to switch the order of the two subscripts in these arrays after each sweep. Since these arrays are used in many other areas of the code, this idea was discarded as being unnecessarily confusing. It should be noted, however, that there are some other arrays in Proteus in which the order of the two subscripts does switch between sweeps. This is described in the next subsection.
\[
\begin{align*}
fl(iiml) &= f(il-1,i2) \\
fl(iii) &= f(il,i2) \\
fl(iipl) &= f(il+1,i2)
\end{align*}
\]

And for the second sweep,
\[
\begin{align*}
fl(iiml) &= f(il,i2-1) \\
fl(iii) &= f(il,i2) \\
fl(iipl) &= f(il,i2+1)
\end{align*}
\]

**Multi-Dimensional Addressing of One-Dimensional Arrays**

As noted in the previous subsection, there are some arrays in *Proteus* in which the order of the first two subscripts does switch between ADI sweeps. These are the A, B, C, and S arrays, which represent the coefficient submatrices and the source term subvector, and the METX, METY, and METT arrays, which represent the metric coefficients in the sweep direction. (A, B, and C are actually four-dimensional arrays, with the third and fourth subscripts representing the equation and dependent variable, respectively. Similarly, S is actually a three-dimensional array, with the third subscript representing the equation. Only the first two subscripts switch between sweeps, however.) For these arrays, the first subscript is the index in the non-sweep direction (i.e., the \( \eta \) direction for the first sweep and the \( \xi \) direction for the second sweep), and the second is the index in the sweep direction (i.e., \( \xi \) for the first sweep and \( \eta \) for the second sweep.)

These multi-dimensional arrays are actually equivalenced to corresponding one-dimensional arrays, stored in common blocks NUM1 and METRIC1. The equivalence is done in subroutine EXEC, which manages the solution of the mean flow equations, and in subroutine EXECT, which manages the solution of the \( k-\epsilon \) turbulence model equations. The multi-dimensional arrays and the appropriate dimensions, which depend on the ADI sweep, are then passed into lower level routines via the argument list. In the lower level routines they can then be referenced as normal, multi-dimensional arrays.

Thus, in subroutine EXEC, we have
\[
\begin{align*}
dimension \quad &a(1,1,1,1), b(1,1,1,1), c(1,1,1,1), s(1,1,1) \\
equivalence \quad & (ama_l(1),a(1,1,1,1)), (bmatl(1), b(1,1,1,1)), \\
& (ematl(1), c(1,1,1,1)), (svectl(1), s(1,1,1)) \\
\end{align*}
\]

Here METX1, METY1, and METT1 are one-dimensional arrays of length \( NIP \times N2P \) stored in common block METRIC1. Similarly, AMAT1, BMAT1, CMAT1, and SVECT1 are one-dimensional arrays stored in common block NUM1. AMAT1, BMAT1, and CMAT1 are of length \( NIP \times N2P \times NEQP \times NEQP \), and SVECT1 is of length \( NIP \times N2P \times NEQP \).

Using COEFC as an example of a lower level routine, we have
\[
\begin{align*}
\text{subroutine coefc} \quad & (a,b,c,s,metx,mety,mett,nvd,nptsd) \\
dimension \quad & a(nvd,nptsd,neqp,neqp), b(nvd,nptsd,neqp,neqp), \\
& c(nvd,nptsd,neqp,neqp), s(nvd,nptsd,neqp) \\
\end{align*}
\]

where NVD and NPTSD are the dimensions in the non-sweep and sweep directions, respectively. For the first sweep, COEFC is thus called from EXEC as
\[
call \text{coefc} \quad (a,b,c,s,metx,mety,mett,n2p,nlp)
\]

And for the second sweep, COEFC is called as

\[\text{An alternative would be use the maximum of N1P and N2P as the size for both of the first two dimensions. In fact, this is what was done in earlier versions of Proteus. However, if N1P is significantly different from N2P, this is inefficient, requiring much more storage than the current procedure.}\]
Two-Level Storage

With the Beam-Warming time differencing scheme used in Proteus, the dependent variables RHO, U, V, W, and ET must be stored at two time levels. For convenience, T is also stored at two time levels. In the ADI solution procedure, RHO, U, etc. are at the known time level \( n \). The corresponding variable at the other time level is denoted by adding an "L" to the variable name. Exactly which time level the "L" variable is at depends on the stage in the solution procedure. Letting \( F \) represent one of these variables, the time levels for \( F \) and \( FL \) are listed in the following table for the different stages of the solution procedure. Recall that * represents the intermediate time level after the first ADI sweep.

<table>
<thead>
<tr>
<th>STAGE IN TIME STEP</th>
<th>TIME LEVEL FOR F</th>
<th>TIME LEVEL FOR FL</th>
</tr>
</thead>
<tbody>
<tr>
<td>From start to end of sweep 1</td>
<td>( n )</td>
<td>( n - 1 )</td>
</tr>
<tr>
<td>From end of sweep 1 to end of sweep 2</td>
<td>( n )</td>
<td>*</td>
</tr>
<tr>
<td>From end of sweep 2 to update in EXEC</td>
<td>( n )</td>
<td>( n + 1 )</td>
</tr>
<tr>
<td>From update in EXEC to start of next step</td>
<td>( n + 1 )</td>
<td>( n )</td>
</tr>
</tbody>
</table>

DUMMY Array

For convenience, a two-dimensional array called DUMMY is stored in common block DUMMY1 and used as a temporary storage location in several areas of the code. This array is dimensioned \( N1P \) by \( N2P \), the same as the flow variables, metrics, etc. DUMMY is used internally in subroutines CONV and RESID. It is also defined in subroutines BLIN2 and BLOUT2 for use in TURBBL, and in subroutine YPLUSN for use in subroutines COEFS1 and KEINIT. And finally, it is defined in subroutine OUTPUT and passed as an argument into subroutine PRTOUT. Details on its use are presented in the subroutine descriptions in Section 4.0.
3.0 COMMON BLOCKS

Transfer of data between routines in Proteus is primarily accomplished through the use of labeled common blocks. Each common block contains variables dealing with a particular aspect of the analysis, and is stored in a separate Cray COMDECK (Cray Research, Inc., 1988). The common block names are the same as the COMDECK names. These names also correspond to the names of the input namelists. All the variables in namelist BC are stored in common block BC1, etc. The Fortran variables in each common block are stored in alphabetical order.

3.1 COMMON BLOCK SUMMARY

<table>
<thead>
<tr>
<th>Block Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC1</td>
<td>Boundary condition parameters for the mean flow equations.</td>
</tr>
<tr>
<td>BC2</td>
<td>Boundary condition parameters for the $k$-$\varepsilon$ equations.</td>
</tr>
<tr>
<td>DUMMY1</td>
<td>Scratch array.</td>
</tr>
<tr>
<td>FLOW1</td>
<td>Variables dealing with fluid properties and the flow being computed.</td>
</tr>
<tr>
<td>GMTRY1</td>
<td>Parameters defining the geometric configuration.</td>
</tr>
<tr>
<td>IC1</td>
<td>Variables needed for setting up initial conditions.</td>
</tr>
<tr>
<td>IO1</td>
<td>Parameters dealing with program input/output requirements.</td>
</tr>
<tr>
<td>METRIC1</td>
<td>Metrics of the nonorthogonal grid transformation, plus the Cartesian coordinates of the grid points.</td>
</tr>
<tr>
<td>NUM1</td>
<td>Parameters associated with the numerical method for the mean flow equations.</td>
</tr>
<tr>
<td>NUM2</td>
<td>Parameters associated with the numerical method for the $k$-$\varepsilon$ equations.</td>
</tr>
<tr>
<td>RSTRRT1</td>
<td>Parameters dealing with the restart option.</td>
</tr>
<tr>
<td>TIME1</td>
<td>Parameters dealing with the time step selection and convergence determination.</td>
</tr>
<tr>
<td>TITLE1</td>
<td>Descriptive title for case being run.</td>
</tr>
<tr>
<td>TURB1</td>
<td>Turbulence parameters.</td>
</tr>
<tr>
<td>TURB20</td>
<td>Parameters and constants associated with the $k$-$\varepsilon$ equations.</td>
</tr>
</tbody>
</table>

3.2 COMMON VARIABLES LISTED ALPHABETICALLY

In this section all the Proteus Fortran variables stored in common blocks are defined, listed alphabetically by variable name. Those marked with an asterisk are input variables. More details on these variables may be found in Section 3.1 of Volume 2. The common block each variable is stored in is given in parentheses at the end of each definition. For subscripted variables, the subscripts are defined along with the variable, except for the subscripts $I_1$ and $I_2$, which are the indices $i$ and $j$ in the $\xi$ and $\eta$ directions, respectively, and run from 1 to $N_1$ and $N_2$. 
This list also includes the parameters used as array dimensions. These are not actually stored in a common block, but are stored in the Cray COMDECK PARAMS1. More details may be found in Section 6.2 of Volume 2.

Unless otherwise noted, all variables representing physical quantities are nondimensional. The nondimensionalizing procedure is described in Section 3.1.1 of Volume 2. The type (real or integer) of the variables follows standard Fortran convention, unless stated otherwise. (I.e., those starting with I, J, K, L, M, or N are integer, and the remainder are real.)

<table>
<thead>
<tr>
<th>Fortran Variable</th>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>See AMAT1. (NUM1)</td>
</tr>
<tr>
<td>AMAT1(I)</td>
<td>A</td>
<td>Subdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. In actual use, this one-dimensional array is equivalenced to the four-dimensional array A(IV,IS,J,K). IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the &quot;vectorized&quot; direction (i.e., the non-sweep direction in which the &quot;BLK&quot; routines are vectorized), and runs from 2 to NEQ - 1. The subscript J = 1 to NEQ, corresponding to the NEQ coupled governing equations, and K = 1 to N, corresponding to the N dependent variables. (NUM1)</td>
</tr>
<tr>
<td>APLUS</td>
<td>A*</td>
<td>Van Driest damping constant in the inner and outer regions of the Baldwin-Lomax turbulence model. (TURB1)</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>See BMAT1. (NUM1)</td>
</tr>
<tr>
<td>BMAT1(I)</td>
<td>B</td>
<td>Diagonal submatrix of coefficients in the block tridiagonal coefficient matrix. In actual use, this one-dimensional array is equivalenced to the four-dimensional array B(IV,IS,J,K). IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the &quot;vectorized&quot; direction (i.e., the non-sweep direction in which the &quot;BLK&quot; routines are vectorized), and runs from 2 to NEQ - 1. The subscript J = 1 to NEQ, corresponding to the NEQ coupled governing equations, and K = 1 to NEQ, corresponding to the NEQ dependent variables. (NUM1)</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>See CMAT1. (NUM1)</td>
</tr>
<tr>
<td>CAVS2E(I)</td>
<td>(\varepsilon^2_2) or (\kappa_2)</td>
<td>Second-order explicit artificial viscosity coefficient in constant coefficient model, or user-specified constant in nonlinear coefficient model. The subscript I = 1 to NEQ, corresponding to the NEQ coupled governing equations. (NUM1)</td>
</tr>
<tr>
<td>CAVS2I(I)</td>
<td>(\varepsilon_I)</td>
<td>Second-order implicit artificial viscosity coefficient in constant coefficient model. The subscript I = 1 to NEQ, corresponding to the NEQ coupled governing equations. (NUM1)</td>
</tr>
<tr>
<td>CAVS4E(I)</td>
<td>(\varepsilon^4_4) or (\kappa_4)</td>
<td>Fourth-order explicit artificial viscosity coefficient in constant coefficient model, or user-specified constant in nonlinear coefficient model. The subscript I = 1 to NEQ, corresponding to the NEQ coupled governing equations. (NUM1)</td>
</tr>
</tbody>
</table>
| CB               | B      | Constant used in the formula for the Klebanoff intermittency factor \(\bar{F}_{kue}\) in the outer region of the Baldwin-Lomax turbu-
* CCLAU $K$ Clauser constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)

* CCP $C_{cp}$ Constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)

CCP1-4 $C_{cp1} - C_{cp2}$ Constants in empirical formula for specific heat as a function of temperature. (FLOW1)

* CFL(I) The ratio $\Delta t/\Delta t_{ref}$ where $\Delta t$ is the actual time step used in the implicit calculation and $\Delta t_{ref}$ is the allowable time step based on the Courant-Friedrichs-Lewy (CFL) criterion for explicit methods. $I$ is the time step sequence number, and runs from 1 to NTSEQ. (TIME1)

* CFLMAX Maximum allowed value of the CFL number. (TIME1)

* CFLMIN Minimum allowed value of the CFL number. (TIME1)

CHGAVG(I) $\Delta Q_{rel}$ Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps. The subscript $I = 1$ to N, corresponding to the $N_\text{eq}$ dependent variables. (TIME1)

CHGMAX(I,J) $\Delta Q_{rel}$ Maximum change in absolute value of the dependent variables over a single time step. The subscript $I = 1$ to $N_\text{eq}$, corresponding to the $N_\text{eq}$ dependent variables, and $J = 1$ to NITAVG, the number of time steps used in the moving average option for determining convergence. (TIME1)

* CHG1 Minimum change, in absolute value, that is allowed in any dependent variable before increasing the time step. (TIME1)

* CHG2 Maximum change, in absolute value, that is allowed in any dependent variable before decreasing the time step. (TIME1)

* CKLEB $C_{Kleb}$ Constant used in the formula for the Klebanoff intermittency factor $F_{Kleb}$ in the outer region of the Baldwin-Lomax turbulence model. (TURB1)

* CKMIN $(C_{Kleb})_{mn}$ Constant used in the formula for the Klebanoff intermittency factor $F_{Kleb}$ in the outer region of the Baldwin-Lomax turbulence model. (TURB1)

CK1-2 $C_{s1} - C_{s2}$ Constants in empirical formula for thermal conductivity coefficient as a function of temperature. (FLOW1)

CMAT1(I) $C$ Superdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. In actual use, this one-dimensional array is equivalenced to the four-dimensional array $C(IV, IS, J, K)$. $IS$ is the grid index in the sweep direction, running from 1 to $N$. $IV$ is the grid index in the "vectorized"

---

5 For the energy equation, the change in $E_T$ is divided by $E_T = \rho \bar{R} T_0 (\gamma - 1) + u^2/2$, so that it is the same order of magnitude as the other conservation variables.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMUR</td>
<td>$C_{\mu}$</td>
<td>Constant used to compute $C_{\mu}$ in the turbulent viscosity formula for the $k$-$\varepsilon$ equations. (TURB20)</td>
</tr>
<tr>
<td>CMU1-2</td>
<td>$C_{\mu1} - C_{\mu2}$</td>
<td>Constants in empirical formula for laminar viscosity coefficient as a function of temperature. (FLOW1)</td>
</tr>
<tr>
<td>CNA</td>
<td>$n$</td>
<td>Exponent in the formula used to average the two outer region $\mu$ profiles that result when both boundaries in a coordinate direction are solid surfaces. (TURB1)</td>
</tr>
<tr>
<td>CNL</td>
<td>$n$</td>
<td>Exponent in the Launder-Priddin modified mixing length formula for the inner region of the Baldwin-Lomax turbulence model. (TURB1)</td>
</tr>
<tr>
<td>CONE</td>
<td>$C_1$</td>
<td>Constant used in the production term of the $\varepsilon$ equation. (TURB20)</td>
</tr>
<tr>
<td>CP(11,12)</td>
<td>$c_p$</td>
<td>Specific heat at constant pressure at time level $n$. (FLOW1)</td>
</tr>
<tr>
<td>CTHREE</td>
<td>$C_3$</td>
<td>Constant used to compute $C_p$ in the turbulent viscosity formula for the $k$-$\varepsilon$ equations. (TURB20)</td>
</tr>
<tr>
<td>CTWOR</td>
<td>$C_2s$</td>
<td>Constant used to compute $C_2$ in the dissipation term of the $\varepsilon$ equation. (TURB20)</td>
</tr>
<tr>
<td>CV(11,12)</td>
<td>$c_v$</td>
<td>Specific heat at constant volume at time level $n$. (FLOW1)</td>
</tr>
<tr>
<td>CVK</td>
<td>$\kappa$</td>
<td>Von Karman mixing length constant used in the inner region of the Baldwin-Lomax and Spalding-Kleinstein turbulence models. (TURB1)</td>
</tr>
<tr>
<td>CWK</td>
<td>$C_{\omega k}$</td>
<td>Constant used in the formula for $F_{\omega k}$, in the outer region of the Baldwin-Lomax turbulence model. (TURB1)</td>
</tr>
<tr>
<td>DEL</td>
<td>$\Delta x$ or $\Delta y$</td>
<td>Computational grid spacing in the ADI sweep direction. (NUM1)</td>
</tr>
<tr>
<td>DETA</td>
<td>$\Delta y$</td>
<td>Computational grid spacing in the $y$ direction. (NUM1)</td>
</tr>
<tr>
<td>DPDETI</td>
<td>$\partial p/\partial E_T$</td>
<td>The derivative of $p$ with respect to $E_T$, stored as a one-dimensional array in the sweep direction. The subscript $I$ therefore runs from 1 to $N$. (FLOW1)</td>
</tr>
<tr>
<td>DPDRHOI</td>
<td>$\partial p/\partial \rho$</td>
<td>The derivative of $p$ with respect to $\rho$, stored as a one-dimensional array in the sweep direction. The subscript $I$ therefore runs from 1 to $N$. (FLOW1)</td>
</tr>
<tr>
<td>DPDRUI</td>
<td>$\partial p/\partial (\rho u)$</td>
<td>The derivative of $p$ with respect to $\rho u$, stored as a one-dimensional array in the sweep direction. The subscript $I$ therefore runs from 1 to $N$. (FLOW1)</td>
</tr>
</tbody>
</table>
DPDRV(I) \( \frac{\partial p}{\partial (\rho v)} \) The derivative of \( p \) with respect to \( \rho v \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). (FLOW1)

DPDRW(I) \( \frac{\partial p}{\partial (\rho w)} \) The derivative of \( p \) with respect to \( \rho w \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). (FLOW1)

\* DT(I) \( \Delta t \) The time step size, when specified directly as input. \( I \) is the time step sequence number, and runs from 1 to \( NTSEQ \). (TIME1)

DTAU(I1,I2) \( \Delta \tau \) Computational time step size. (TIME1)

DTDET(I) \( \frac{\partial T}{\partial E_T} \) The derivative of \( T \) with respect to \( E_T \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). (FLOW1)

DTDRHO(I) \( \frac{\partial T}{\partial \rho} \) The derivative of \( T \) with respect to \( \rho \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). (FLOW1)

DTDRU(I) \( \frac{\partial T}{\partial (\rho u)} \) The derivative of \( T \) with respect to \( \rho u \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). (FLOW1)

DTDRV(I) \( \frac{\partial T}{\partial (\rho v)} \) The derivative of \( T \) with respect to \( \rho v \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). (FLOW1)

DTDRW(I) \( \frac{\partial T}{\partial (\rho w)} \) The derivative of \( T \) with respect to \( \rho w \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). (FLOW1)

\* DTF1 Factor by which the time step is multiplied if the solution changes too slowly. (TIME1)

\* DTF2 Factor by which the time step is divided if the solution changes too quickly. (TIME1)

\* DTMAX Maximum value that \( \Delta \tau \) is allowed to reach, or the maximum \( \Delta \tau \) used in the time step cycling procedure. (TIME1)

\* DTMIN Minimum value that \( \Delta \tau \) is allowed to reach, or the minimum \( \Delta \tau \) used in the time step cycling procedure. (TIME1)

DUMMY(I1,I2) Dummy array used for temporary storage in several subroutines. (DUMMY1)

DXI \( \Delta \xi \) Computational grid spacing in the \( \xi \) direction. (NUM1)

E(I1,I2) \( \epsilon \) Turbulent dissipation rate at time level \( n \). (TURB20)

EL(I1,I2) \( \epsilon \) Turbulent dissipation rate at previous or intermediate time level. (TURB20)

\* EPS(I) \( \epsilon \) Convergence level to be reached. The subscript \( I \) runs from 1 to \( N_q \), corresponding to the \( N_q \) dependent variables. (TIME1)
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER</td>
<td>Dimensional reference energy, $p\cdot u^2$. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>ET(I1,I2)</td>
<td>Total energy at time level $n$. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>ETAT(I1,I2)</td>
<td>The derivative of the computational coordinate $\eta$ with respect to untransformation time $t$. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>ETAX(I1,I2)</td>
<td>The derivative of the computational coordinate $\eta$ with respect to the Cartesian coordinate $x$. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>ETAY(I1,I2)</td>
<td>The derivative of the computational coordinate $\eta$ with respect to the Cartesian coordinate $y$ or cylindrical coordinate $r$. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>ETL(I1,I2)</td>
<td>Total energy at previous or intermediate time level. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>* FBCT1(I2,J)</td>
<td>Point-by-point values used for boundary conditions for the $k$-$\varepsilon$ turbulence model on the $\xi = 0$ and $\xi = 1$ boundaries. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. The subscript $I = 1$ or 2, corresponding to the $k$ and $\varepsilon$ equations, respectively, and $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)</td>
<td></td>
</tr>
<tr>
<td>* FBCT2(I2,J)</td>
<td>Point-by-point values used for boundary conditions for the $k$-$\varepsilon$ turbulence model on the $\eta = 0$ and $\eta = 1$ boundaries. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. The subscript $I = 1$ or 2, corresponding to the $k$ and $\varepsilon$ equations, respectively, and $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)</td>
<td></td>
</tr>
<tr>
<td>* FBC1(I2,J)</td>
<td>Point-by-point values used for steady boundary conditions on the $\xi = 0$ and $\xi = 1$ surfaces. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. $I$ runs from 1 to $V_q$, corresponding to the $N_q$ conditions needed, and $J = 1$ or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)</td>
<td></td>
</tr>
<tr>
<td>* FBC2(I2,J)</td>
<td>Point-by-point values used for steady boundary conditions on the $\eta = 0$ and $\eta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is being specified by the user, or by the program itself. $I$ runs from 1 to $V_q$, corresponding to the $N_q$ conditions needed, and $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)</td>
<td></td>
</tr>
<tr>
<td>* GAMR</td>
<td>Reference ratio of specific heats, $c_p/c_r$. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>* GBCT1(I,J)</td>
<td>Values used for boundary conditions for the $k$-$\varepsilon$ turbulence model on the $\xi = 0$ and $\xi = 1$ boundaries, when specified for the entire surface. The subscript $I = 1$ or 2, corresponding to the $k$ and $\varepsilon$ equations, respectively, and $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)</td>
<td></td>
</tr>
<tr>
<td>* GBCT2(I,J)</td>
<td>Values used for boundary conditions for the $k$-$\varepsilon$ turbulence model on the $\eta = 0$ and $\eta = 1$ boundaries, when specified for the entire surface. The subscript $I = 1$ or 2, corresponding to the $k$ and $\varepsilon$ equations, respectively, and $J = 1$ or 2, corre-</td>
<td></td>
</tr>
</tbody>
</table>
**GBC1(I,J)**

Values used for steady boundary conditions on the \( \zeta = 0 \) and \( \zeta = 1 \) boundaries, when specified for the entire surface. \( I \) runs from 1 to \( N_q \), corresponding to the \( N_q \) conditions needed, and \( J = 1 \) or 2, corresponding to the \( \zeta = 0 \) and \( \zeta = 1 \) boundaries, respectively. (BC1)

**GBC2(I,J)**

Values used for steady boundary conditions on the \( \eta = 0 \) and \( \eta = 1 \) boundaries, when specified for the entire surface. \( I \) runs from 1 to \( N_q \), corresponding to the \( N_q \) conditions needed, and \( J = 1 \) or 2, corresponding to the \( \eta = 0 \) and \( \eta = 1 \) boundaries, respectively. (BC2)

**GC**

Dimensional proportionality factor in Newton’s second law, either 32.174 lbm-ft/lb-sec\(^2\) or 1.0 kg-m/N-sec\(^2\). (FLOW1)

**GTBC1(K,I,J)**

A variable used to specify the values for unsteady and time-periodic boundary conditions on the \( \zeta = 0 \) and \( \zeta = 1 \) boundaries. \( I \) runs from 1 to \( N_q \), corresponding to the \( N_q \) conditions needed, and \( J = 1 \) or 2, corresponding to the \( \zeta = 0 \) and \( \zeta = 1 \) boundaries, respectively. For general unsteady boundary conditions, \( K = 1 \) to NTBC, corresponding to the time levels in the array NTBCA, and GTBC1 specifies the boundary condition value directly. For time-periodic boundary conditions, \( K = 1 \) to 4, and GTBC1 specifies the four coefficients in the equation used to determine the boundary condition value. (BC1)

**GTBC2(K,I,J)**

A variable used to specify the values for unsteady and time-periodic boundary conditions on the \( \eta = 0 \) and \( \eta = 1 \) boundaries. \( I \) runs from 1 to \( N_q \), corresponding to the \( N_q \) conditions needed, and \( J = 1 \) or 2, corresponding to the \( \eta = 0 \) and \( \eta = 1 \) boundaries, respectively. For general unsteady boundary conditions, \( K = 1 \) to NTBC, corresponding to the time levels in the array NTBCA, and GTBC2 specifies the boundary condition value directly. For time-periodic boundary conditions, \( K = 1 \) to 4, and GTBC2 specifies the four coefficients in the equation used to determine the boundary condition value. (BC1)

**HSTAG**

Stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)

**HSTAGR**

Dimensional stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)

**IAV2E**

Flag for second-order explicit artificial viscosity; 0 for none, 1 for constant coefficient model, 2 for nonlinear coefficient model. (NUM1)

**IAV2I**

Flag for second-order implicit artificial viscosity; 0 for none, 1 for constant coefficient model. (NUM1)

**IAV4E**

Flag for fourth-order explicit artificial viscosity; 0 for none, 1 for constant coefficient model, 2 for nonlinear coefficient model. (NUM1)
- **IBASE**
  Base index used with ISTEP to compute one-dimensional index for two-dimensional array. Then, for example, for any sweep U(I1,I2) = U1(IBASE + ISTEP*(I - 1)) where I is the grid index in the sweep direction. (NUM1)

- **IBCELMM(I,J)**
  Flags for elimination of off-diagonal sub-matrices resulting from gradient or extrapolation boundary conditions: 0 if elimination is not necessary, 1 if it is. The subscript I = 1 or 2 corresponding to the sweep direction, and J = 1 or 2 corresponding to the lower or upper boundary in that direction. (BC1)

- **IBCT1(I2,I,J)**
  Flags specifying, point-by-point, the type of boundary conditions used for the $k$-$\varepsilon$ turbulence model on the $\xi = 0$ and $\xi = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the $k$ and $\varepsilon$ equations, respectively, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC2)

- **IBCT2(I1,I,J)**
  Flags specifying, point-by-point, the type of boundary conditions used for the $k$-$\varepsilon$ turbulence model on the $\eta = 0$ and $\eta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. The subscript I = 1 or 2, corresponding to the $k$ and $\varepsilon$ equations, respectively, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC2)

- **IBC1(I2,I,J)**
  Flags specifying, point-by-point, the type of steady boundary conditions used on the $\xi = 0$ and $\xi = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. I runs from 1 to $N_x$, corresponding to the $N_x$ conditions needed, and J = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)

- **IBC2(I1,I,J)**
  Flags specifying, point-by-point, the type of steady boundary conditions used on the $\eta = 0$ and $\eta = 1$ surfaces. These are either set in the input, if a point-by-point distribution is specified by the user, or by the program itself. I runs from 1 to $N_y$, corresponding to the $N_y$ conditions needed, and J = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)

- **IBVUP(I)**
  Flags for updating boundary values from the first sweep after the last sweep: 0 if updating is not necessary, 1 if it is. Updating is required when gradient or extrapolation boundary conditions are used. The subscript I = 1 or 2, corresponding to the lower or upper boundary in the first sweep direction. (BC1)

- **ICHECK**
  Results are checked for convergence every ICHECK'th time level. (TIME1)

- **ICONV**
  Convergence flag; 0 if not converged, 1 if converged. (TIME1)
• ICTEST  Flag for convergence criteria to be used. (TIME1)
• ICVARS Parameter specifying which variables are being supplied as initial conditions by subroutine INIT. (FLOW1)
• IDEBUG(I)  A 20-element array of flags specifying various debug options. (IO1)
• IDTAU  Flag for time step selection method. (TIME1)
• IDTMOD  The time step size is modified every IDTMOD'th time step. (TIME1)
• IEULER  Flag for Euler calculation option; 0 for a full time-averaged Navier-Stokes calculation, 1 for an Euler calculation. (FLOW1)

IGAM  Flag set by method used to select GAMR; 0 if GAMR is defaulted (and hence $c_p$ and $c_v$ are functions of temperature), 1 if GAMR is specified by user (and hence $c_p$ and $c_v$ are constants). (FLOW1)

• IGINT(I)  Flags for grid interpolation requirement; 0 if interpolation is not needed, 1 if interpolation is needed. The subscript $I = 1$ or 2, corresponding to the $\xi$ and $\eta$ directions, respectively. (GMTRY1)

• IHSTAG  Flag for constant stagnation enthalpy option; 0 to solve the energy equation, 1 to eliminate the energy equation by assuming constant stagnation enthalpy. (FLOW1)

• ILAMV  Flag for computation of laminar viscosity and thermal conductivity; 0 for constant values, 1 for functions of local temperature. (FLOW1)

• ILDAMP  Flag for the Launder-Priddin modified mixing length formula in the inner region of the Baldwin-Lomax turbulence model. (TURB1)

• INEG  Flag indicating non-positive values of pressure and/or temperature: 0 for no non-positive values, 1 for some. (FLOW1)

• INNER  Flag for type of inner region turbulence model. (TURB1)

• IPACK(I)  Flags for grid packing option; 0 for no packing, 1 to pack points as specified by the input array SQ. The subscript $I = 1$ or 2, corresponding to the $\xi$ and $\eta$ directions, respectively. (NUM1)

• IPILOT  Flag controlling the creation of an auxiliary file, usually called a "plot file", used for later post-processing. (IO1)

• IPLT  Results are written into the plot file every IPLT time levels. (IO1)

• IPLTA(I)  Time levels at which results are written into the plot file. The subscript $I = 1$ to 101, the maximum number of time levels that may be written. (IO1)
- **IPRT**
  Results are printed every IPRT time levels. (IO1)

- **IPRTA(I)**
  Time levels at which results are printed. The subscript I = 1 to 101, the maximum number of time levels that may be printed. (IO1)

- **IPRT1**
  Results are printed at every IPRT1'th mesh point in the $\xi$ direction. (IO1)

- **IPRT2**
  Results are printed at every IPRT2'th mesh point in the $\eta$ direction. (IO1)

- **IPRT1A(I)**
  $\xi$ indices at which results are printed. The subscript I = 1 to a maximum of N1, the number of grid points in the $\xi$ direction. (IO1)

- **IPRT2A(I)**
  $\eta$ indices at which results are printed. The subscript I = 1 to a maximum of N2, the number of grid points in the $\eta$ direction. (IO1)

- **IREST**
  Flag controlling the reading and writing of auxiliary files used for restarting the calculation in a separate run. (RSTRT1)

- **ISTEP**
  Multiplication factor used with IBASE to compute one-dimensional index for two-dimensional array. (NUM1)

- **ISWEEP**
  Flag specifying ADI sweep direction; 1 for $\xi$ direction and 2 for $\eta$ direction. (NUM1)

- **ISWIRL**
  Flag for swirl in axisymmetric flow; 0 for no swirl, 1 for swirl. (FLOW1)

- **IT**
  Current time step number, or known time level. Time step number $n$ updates the solution from time level $n$ to $n+1$. (TIME1)

- **ITBEG**
  The time step number, or known time level $n$, at the beginning of a run. For a non-restart case, ITBEG = 1. (TIME1)

- **ITDBC**
  Flag for time-dependent boundary conditions; 0 if all boundary conditions are steady, 1 if any general unsteady boundary conditions are used, 2 if only steady and time-periodic boundary conditions are used. (BC1)

- **ITEND**
  The final time step number. (TIME1)

- **ITETA**
  Flag for computing turbulent viscosity on constant $\eta$ lines. (TURB1)

- **ITHIN(I)**
  Flags for thin-layer option; 0 to include 2nd. derivative viscous terms, 1 to eliminate them. The subscript I = 1 or 2, corresponding to the $\xi$ and $\eta$ directions, respectively. (FLOW1)

- **ITSEQ**
  Current time step sequence number. (TIME1)

- **ITURB**
  Flag for turbulent flow option; 0 for laminar flow, 1 for turbulent flow using the Baldwin-Lomax algebraic turbulence model. (FLOW1)
model, 20 for turbulent flow using the Chien two-equation k-ε turbulence model. (TURB1)

* ITXI
  Flag for computing turbulent viscosity on constant \( \xi \) lines. (TURB1)

* IUNITS
  Flag for type of units used to specify reference conditions; 0 for English units, 1 for SI units. (IO1)

IV \( i \)
  Grid point index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized). Therefore, IV = j for the first sweep and i for the second sweep. (NUM1)

* IVOUT(I)
  A 50-element array specifying which variables are to be printed. (IO1)

* IWALL1(I)
  Flags indicating type of surfaces in the \( \xi \) direction; 0 for a free boundary, 1 for a solid wall. The subscript I = 1 or 2, corresponding to the \( \xi = 0 \) and \( \xi = 1 \) surfaces, respectively. (TURB1)

* IWALL2(I)
  Flags indicating type of surfaces in the \( \eta \) direction; 0 for a free boundary, 1 for a solid wall. The subscript I = 1 or 2, corresponding to the \( \eta = 0 \) and \( \eta = 1 \) surfaces, respectively. (TURB1)

* IWOUT1(I)
  Flags specifying whether or not various parameters are to be printed along the \( \xi \) boundaries; 0 for no printout, 1 for printout along the boundary. The subscript I = 1 or 2, corresponding to the \( \xi = 0 \) and \( \xi = 1 \) boundaries, respectively. (IO1)

* IWOUT2(I)
  Flags specifying whether or not various parameters are to be printed along the \( \eta \) boundaries; 0 for no printout, 1 for printout along the boundary. The subscript I = 1 or 2, corresponding to the \( \eta = 0 \) and \( \eta = 1 \) boundaries, respectively. (IO1)

I1 \( i \)
  Grid point index in the \( \xi \) direction. (NUM1)

I2 \( j \)
  Grid point index in the \( \eta \) direction. (NUM1)

* JBCT1(I,J)
  Flags specifying the type of boundary conditions used for the k-ε turbulence model on the \( \xi = 0 \) and \( \xi = 1 \) surfaces, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the \( \xi = 0 \) and \( \xi = 1 \) boundaries, respectively. (BC2)

* JBCT2(I,J)
  Flags specifying the type of boundary conditions used for the k-ε turbulence model on the \( \eta = 0 \) and \( \eta = 1 \) surfaces, when specified for the entire surface. The subscript I = 1 or 2, corresponding to the k and ε equations, respectively, and J = 1 or 2, corresponding to the \( \eta = 0 \) and \( \eta = 1 \) boundaries, respectively. (BC2)

* JBC1(I,J)
  Flags specifying the type of steady boundary conditions used on the \( \xi = 0 \) and \( \xi = 1 \) surfaces, when specified for the entire
surface. I runs from 1 to $N_{\eta}$, corresponding to the $N_{\eta}$ conditions needed, and $J = 1$ or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)

- **JBC2(I,J)**

  Flags specifying the type of steady boundary conditions used on the $\eta = 0$ and $\eta = 1$ surfaces, when specified for the entire surface. I runs from 1 to $N_{\eta}$, corresponding to the $N_{\eta}$ conditions needed, and $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)

- **JI(I1,I2)**

  Normally the inverse Jacobian of the non-orthogonal grid transformation. For the COEF routines in axisymmetric flow, it is temporarily redefined as the product of the local radius and the inverse Jacobian. This is a real variable. (METRIC1)

- **JTBC1(I,J)**

  A variable specifying the type of time dependency for the boundary conditions on the $\xi = 0$ and $\xi = 1$ boundaries. I runs from 1 to $N_{\eta}$, corresponding to the $N_{\eta}$ conditions needed, and $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)

- **JTBC2(I,J)**

  A variable specifying the type of time dependency for the boundary conditions on the $\eta = 0$ and $\eta = 1$ boundaries. I runs from 1 to $N_{\eta}$, corresponding to the $N_{\eta}$ conditions needed, and $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)

- **KBCPER(I)**

  Flags for spatially periodic boundary conditions: 0 for non-periodic, I for periodic. The subscript $I = 1$ or 2, corresponding to the $\xi$ and $\eta$ directions, respectively. (BC1)

- **KBC1(J)**

  Flags for type of boundaries in the $\xi$ direction. The subscript $J = 1$ or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. (BC1)

- **KBC2(J)**

  Flags for type of boundaries in the $\eta$ direction. The subscript $J = 1$ or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. (BC1)

- **KE(I1,I2)**

  Turbulent kinetic energy at time level $n$. This is a real variable. (TURB20)

- **KEL(I1,I2)**

  Turbulent kinetic energy at previous or intermediate time level. This is a real variable. (TURB20)

- **KT(I1,I2)**

  Effective thermal conductivity coefficient at time level $n$. This is a real variable. (FLOW1)

- **KTR**

  Dimensional reference thermal conductivity coefficient. This is a real variable. (FLOW1)

- **LA(I1,I2)**

  Effective second coefficient of viscosity at time level $n$ (usually assumed equal to $-2\mu/3$). This is a real variable. (FLOW1)

- **LR**

  Dimensional reference length. This is a real variable. (FLOW1)
LRMAX(I,J,K)  The grid indices corresponding to the location of the maximum absolute value of the residual. The subscript I = 1 or 2, corresponding to the \( \xi \) and \( \eta \) directions, respectively, J = 1 to \( N_x \), corresponding to the \( N_x \) coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)

LWAKE1(I2)  Grid point index in the \( \xi \) direction used as the origin for computing length scales for free turbulent flows. (TURB1)

LWAKE2(I1)  Grid point index in the \( \eta \) direction used as the origin for computing length scales for free turbulent flows. (TURB1)

* LWALL1(I2,I)  Flags indicating, point-by-point, the type of surfaces in the \( \xi \) direction; 0 for a free boundary, 1 for a solid wall. The subscript I = 1 or 2, corresponding to the \( \xi = 0 \) and \( \xi = 1 \) boundaries, respectively. (TURB1)

* LWALL2(I1,I)  Flags indicating, point-by-point, the type of surfaces in the \( \eta \) direction; 0 for a free boundary, 1 for a solid wall. The subscript I = 1 or 2, corresponding to the \( \eta = 0 \) and \( \eta = 1 \) boundaries, respectively. (TURB1)

LWSET(I)  Flags specifying how wall locations are determined for the turbulence model; 0 if wall locations are found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters. The subscript I = 1 to 4, corresponding to the \( \xi = 0, \xi = 1, \eta = 0, \) and \( \eta = 1 \) boundaries, respectively. (TURB1)

* MACHR  Reference Mach number, \( u_j(y,\bar{R}T)^{1/2} \). This is a real variable. (FLOW1)

METT1(I)  \( \xi \), or \( \eta \), The derivative of the computational coordinate in the ADI sweep direction with respect to untransformed time \( t \). In actual use, this one-dimensional array is equivalenced to the two-dimensional array METT(IV,IS). IS is the grid index in the sweep direction, running from 1 to \( N \). IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N_r - 1 \). This is a real variable. (METRIC1)

METX1(I)  \( \xi \), or \( \eta \), The derivative of the computational coordinate in the ADI sweep direction with respect to the Cartesian coordinate \( x \). In actual use, this one-dimensional array is equivalenced to the two-dimensional array METX(IV,IS). IS is the grid index in the sweep direction, running from 1 to \( N \). IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N_r - 1 \). This is a real variable. (METRIC1)

METY1(I)  \( \xi \), or \( \eta \), The derivative of the computational coordinate in the ADI sweep direction with respect to the Cartesian coordinate \( y \) or cylindrical coordinate \( r \). In actual use, this one-dimensional array is equivalenced to the two-dimensional array METY(IV,IS). IS is the grid index in the sweep direction, running from 1 to \( N \). IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK"
The routines are vectorized, and runs from 2 to $N_t - 1$. This is a real variable. (METRIC1)

- **MU(I1,I2)** $\mu$  
  Effective viscosity coefficient at time level $n$. This is a real variable. (FLOW1)

- **MUR** $\mu_r$  
  Dimensional reference viscosity coefficient. This is a real variable. (FLOW1)

- **MUT(I1,I2)** $\mu_t$  
  Turbulent viscosity coefficient at time level $n$. This is a real variable. (FLOW1)

- **MUTL(I1,I2)** $\mu_l$  
  Turbulent viscosity coefficient at previous or intermediate time level. This is a real variable. (TURB20)

- **NAMAX**  
  A dimensioning parameter equal to the maximum number of time steps allowed in the moving average convergence test (the ICTEST = 2 option). (PARAMS1)

- **NBC**  
  A dimensioning parameter equal to the number of boundary conditions per equation. (PARAMS1)

- **NC**  
  Array index associated with the continuity equation. (NUM1)

- **NDTCYC**  
  Number of time steps per cycle used in the time step cycling procedure. (TIME1)

- **NEN**  
  Array index associated with the energy equation. (NUM1)

- **NEQ** $N_{et}$  
  The number of coupled governing equations actually being solved. (NUM1)

- **NEQP**  
  A dimensioning parameter equal to the number of coupled equations allowed. (PARAMS1)

- **NEQPM**  
  A dimensioning parameter equal to the maximum number of coupled equations available. (PARAMS1)

- **NET**  
  Array index associated with the dependent variable $E_t$. (NUM1)

- **NGEOM**  
  Flag used to specify type of computational coordinates; 1 for Cartesian $(x,y)$ coordinates, 2 for polar $(r',\theta')$ coordinates, and 10 to read the coordinates from unit NGRID. (GMTRY1)

- **NGRID**  
  Unit number for reading grid file. (IO1)

- **NHIST**  
  Unit number for writing convergence history file. (IO1)

- **NHMAX**  
  Maximum number of time levels allowed in the printout of the convergence history file (not counting the first two, which are always printed.) (IO1)

- **NIN**  
  Unit number for reading namelist input. (IO1)

- **NITAVG**  
  Number of time steps used in the moving average convergence test. (TIME1)
NMAXP

* NOUT
  Unit number for writing standard output. (IO1)

* NPLOT
  Unit number for writing CONTOUR or PLOT3D Q plot file. (IO1)

* NPLOTX
  Unit number for writing PLOT3D XYZ plot file. (IO1)

NPRT1
  Total number of indices for printout in the \( \xi \) direction. (IO1)

NPRT2
  Total number of indices for printout in the \( \eta \) direction. (IO1)

NPTS
  The number of grid points in the sweep direction. (NUM1)

NPT1
  \( N_1 \) or \( N_1 + 1 \)
  The number of grid points in the \( \xi \) direction used in computing coefficients: \( N_1 \) for non-periodic boundary conditions; \( N_1 + 1 \) for spatially periodic boundary conditions. (NUM1)

NPT2
  \( N_2 \) or \( N_2 + 1 \)
  The number of grid points in the \( \eta \) direction used in computing coefficients: \( N_2 \) for non-periodic boundary conditions; \( N_2 + 1 \) for spatially periodic boundary conditions. (NUM1)

NR
  Array index associated with the dependent variable \( \rho \). (NUM1)

* NRQIN
  Unit number for reading restart flow field. (RSTRT1)

* NRQOUT
  Unit number for writing restart flow field. (RSTRT1)

NRU
  Array index associated with the dependent variable \( \rho u \). (NUM1)

NRV
  Array index associated with the dependent variable \( \rho v \). (NUM1)

NRW
  Array index associated with the dependent variable \( \rho w \). (NUM1)

* NRXIN
  Unit number for reading restart computational mesh. (RSTRT1)

* NRXOUT
  Unit number for writing restart computational mesh. (RSTRT1)

NSCR1
  Unit number for scratch file in subroutine PLOT. (IO1)

* NTBC
  Number of values in the tables of GTBC1 and/or GTBC2 vs. NTBCA for general unsteady boundary conditions. (BC1)

* NTBCA(I)
  Time levels at which GTBC1 and/or GTBC2 are specified for general unsteady boundary conditions. The subscript \( I = 1 \) to NTBC, corresponding to the NTBC values in the table. (BC1)

* NTIME(I)
  Maximum number of time steps to march. I runs from 1 to NTSEQP, corresponding to the time step sequence number. (TIME1)
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTKE</td>
<td>Number of $k$-$\varepsilon$ iterations per mean flow iteration. (TURB20)</td>
<td></td>
</tr>
<tr>
<td>NTOTP</td>
<td>A dimensioning parameter equal to the total storage required for a single two-dimensional array (i.e., $N1P \times N2P$). (PARAMS1)</td>
<td></td>
</tr>
<tr>
<td>NTP</td>
<td>A dimensioning parameter equal to the maximum number of entries in the table of time-dependent boundary condition values. (PARAMS1)</td>
<td></td>
</tr>
<tr>
<td>NTSEQ</td>
<td>The total number of time step sequences being used. (TIME1)</td>
<td></td>
</tr>
<tr>
<td>NTSEQP</td>
<td>A dimensioning parameter equal to the maximum number of time step sequences in the time step sequencing option. (PARAMS1)</td>
<td></td>
</tr>
<tr>
<td>NV</td>
<td>The number of grid points in the &quot;vectorized&quot; direction (i.e., the non-sweep direction in which the &quot;BLK&quot; routines are vectorized). Therefore, $NV = N2$ for the first sweep and $N1$ for the second sweep. (NUM1)</td>
<td></td>
</tr>
<tr>
<td>NXM</td>
<td>Array index associated with the $x$-momentum equation. (NUM1)</td>
<td></td>
</tr>
<tr>
<td>NYM</td>
<td>Array index associated with the $y$ or $r$-momentum equation. (NUM1)</td>
<td></td>
</tr>
<tr>
<td>NZM</td>
<td>Array index associated with the swirl momentum equation. (NUM1)</td>
<td></td>
</tr>
<tr>
<td>N1</td>
<td>The number of grid points in the $\xi$ direction. (NUM1)</td>
<td></td>
</tr>
<tr>
<td>N1P</td>
<td>A dimensioning parameter equal to the maximum number of grid points in the $\xi$ direction. (PARAMS1)</td>
<td></td>
</tr>
<tr>
<td>N2</td>
<td>The number of grid points in the $\eta$ direction. (NUM1)</td>
<td></td>
</tr>
<tr>
<td>N2P</td>
<td>A dimensioning parameter equal to the maximum number of grid points in the $\eta$ direction. (PARAMS1)</td>
<td></td>
</tr>
<tr>
<td>P(11,12)</td>
<td>Static pressure at time level $n$. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>PONE</td>
<td>Part 1 of the production rate of the turbulent kinetic energy. (TURB20)</td>
<td></td>
</tr>
<tr>
<td>PR</td>
<td>Dimensional reference static pressure, $\rho \bar{R} T_c / g_c$. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>PR1</td>
<td>Reference laminar Prandtl number, $c_p \mu / k$. where $c_p = \gamma R (\gamma - 1)$. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>PRR</td>
<td>Reference Prandtl number, $\mu \omega / k T_r$. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>PRT</td>
<td>Turbulent Prandtl number, or, if non-positive, a flag indicating the use of a variable turbulent Prandtl number. (TURB1)</td>
<td></td>
</tr>
<tr>
<td>PTWO</td>
<td>Part 2 of the production rate of the turbulent kinetic energy. (TURB20)</td>
<td></td>
</tr>
</tbody>
</table>
• P0 $\rho_0$ Initial static pressure. (IC1)

RAX(I) $1$ or $r$ 1 for two-dimensional planar flow, and the local radius $r$ for axisymmetric flow. $I$ is the grid index in the sweep direction, running from 1 to $N$. (METRIC1)

• RER $Re$ Reference Reynolds number, $\rho u L/\mu$. (FLOW1)

RESAVG(J,K) $R_{av}$ The average absolute value of the residual for the previous time step. The subscript $J = 1$ to $N_x$, corresponding to the $N_x$ coupled governing equations, and $K = 1$ or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)

RESL2(J,K) $R_{L2}$ The $L_2$ norm of the residual for the previous time step. The subscript $J = 1$ to $N_x$, corresponding to the $N_x$ coupled governing equations, and $K = 1$ or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)

RESMAX(J,K) $R_{max}$ The maximum absolute value of the residual for the previous time step. The subscript $J = 1$ to $N_x$, corresponding to the $N_x$ coupled governing equations, and $K = 1$ or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)

• REXT1 $Re_{r'}$ Reynolds number at the beginning of the transition region, based on maximum total velocity and distance from $\xi = 0$, for flow predominantly in the $\xi$ direction with a leading edge at $\xi = 0$. (TURB1)

• REXT2 $Re_{r'}$ Reynolds number at the beginning of the transition region, based on maximum total velocity and distance from $\eta = 0$, for flow predominantly in the $\eta$ direction with a leading edge at $\eta = 0$. (TURB1)

• RG $\bar{R}$ Dimensional gas constant. (FLOW1)

RGAS $R$ Nondimensional gas constant. (FLOW1)

RHO(I1,I2) $\rho$ Static density at time level $n$. (FLOW1)

RHOL(I1,I2) $\rho$ Static density at previous or intermediate time level. (FLOW1)

• RHOR $\rho_r$ Dimensional reference density. (FLOW1)

• RMAX $r'_{max}$ Maximum $r'$ coordinate for polar grid option. (GMTRY1)

• RMIN $r'_{min}$ Minimum $r'$ coordinate for polar grid option. (GMTRY1)

S $S$ See SVECT1. (NUM1)

• SIGE $\sigma_e$ Constant used in the diffusion term of the $\varepsilon$ equation. (TURB20)

• SIGK $\sigma_k$ Constant used in the diffusion term of the $k$ equation. (TURB20)
* SQ(I,J)  
An array controlling the packing of grid points using the Roberts transformation. The subscript I = 1 or 2, corresponding to the \( \xi \) and \( \eta \) directions, respectively. SQ(I,1) specifies the location of packing, and SQ(I,2) specifies the amount of packing. (NUM1)

* SVECT1(I)  
S  
Subvector of source terms in the block tridiagonal system of equations. In actual use, this one-dimensional array is equivalent to the three-dimensional array S(IV,IS,J). IS is the grid index in the sweep direction, running from 1 to \( N \). IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N_e - 1 \). The subscript J = 1 to \( N_e \), corresponding to the \( N_e \), coupled governing equations. (NUM1)

* T(I1,I2)  
T  
Static temperature at time level \( n \). (FLOW1)

* TAU(I1,I2)  
\( \tau \)  
Current value of the time marching parameter. (TIME1)

* TFACT  
Factor used in computing the \( k-e \) time step, \( \Delta \tau_{k-e} = TFACT(\Delta \tau) \). (TURB20)

* THC(I)  
\( \theta_1, \theta_2 \)  
A two-element array specifying the time difference centering parameters used for the continuity equation. (NUM1)

* THE(I)  
\( \theta_1, \theta_2, \theta_3 \)  
A three-element array specifying the time difference centering parameters used for the energy equation. (NUM1)

* THKE(I)  
\( \theta_1, \theta_2 \)  
A two-element array specifying the time difference centering parameters used for the \( k-e \) equations. (NUM2)

* THMAX  
\( \theta^{'\max} \)  
Maximum \( \theta' \) coordinate in degrees for polar grid option. (GMTRY1)

* THMIN  
\( \theta^{'\min} \)  
Minimum \( \theta' \) coordinate in degrees for polar grid option. (GMTRY1)

* THX(I)  
\( \theta_1, \theta_2, \theta_3 \)  
A three-element array specifying the time difference centering parameters used for the \( x \)-momentum equation. (NUM1)

* THY(I)  
\( \theta_1, \theta_2, \theta_3 \)  
A three-element array specifying the time difference centering parameters used for the \( y \) or \( r \)-momentum equation. (NUM1)

* THZ(I)  
\( \theta_1, \theta_2, \theta_3 \)  
A three-element array specifying the time difference centering parameters used for the swirl momentum equation. (NUM1)

* TITLE  
Title for printed output and CONTOUR plot file, up to 72 characters long. This is a character variable. (TITLE1)

* TL(I1,I2)  
T  
Static temperature at previous or intermediate time level. (FLOW1)

* TLIM  
When the amount of CPU time remaining for the job drops below TLIM seconds, the calculation is stopped. (TIME1)

* TR  
\( T_r \)  
Dimensional reference temperature. (FLOW1)

* T0  
\( T_0 \)  
Initial static temperature. (IC1)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>U(I,J)</td>
<td>Velocity in the Cartesian x direction at time level n. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>UL(I,J)</td>
<td>Velocity in the Cartesian x direction at previous or intermediate time level. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>UR</td>
<td>Dimensional reference velocity. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>U0</td>
<td>Initial velocity in the Cartesian x direction. (IC1)</td>
<td></td>
</tr>
<tr>
<td>V(I,J)</td>
<td>Velocity in the Cartesian y direction or cylindrical r direction at time level n. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>VL(I,J)</td>
<td>Velocity in the Cartesian y direction or cylindrical r direction at previous or intermediate time level. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>VORT(I,J)</td>
<td>Total vorticity magnitude. (TURB1)</td>
<td></td>
</tr>
<tr>
<td>VORT(I,J)</td>
<td>Production rate of turbulent kinetic energy. (TURB1)</td>
<td></td>
</tr>
<tr>
<td>V0</td>
<td>Initial velocity in the Cartesian y direction or cylindrical r direction. (IC1)</td>
<td></td>
</tr>
<tr>
<td>W(I,J)</td>
<td>Swirl velocity at time level n. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>WL(I,J)</td>
<td>Swirl velocity at previous or intermediate time level. (FLOW1)</td>
<td></td>
</tr>
<tr>
<td>W0</td>
<td>Initial swirl velocity. (IC1)</td>
<td></td>
</tr>
<tr>
<td>X(I,J)</td>
<td>Cartesian x coordinate. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>XIT(I,J)</td>
<td>The derivative of the computational coordinate ξ with respect to untransformed time t. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>XIX(I,J)</td>
<td>The derivative of the computational coordinate ξ with respect to the Cartesian coordinate x. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>XIY(I,J)</td>
<td>The derivative of the computational coordinate ξ with respect to the Cartesian coordinate y or cylindrical coordinate r. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>XMAX</td>
<td>Maximum x coordinate for Cartesian grid option. (GMTRY1)</td>
<td></td>
</tr>
<tr>
<td>XMIN</td>
<td>Minimum x coordinate for Cartesian grid option. (GMTRY1)</td>
<td></td>
</tr>
<tr>
<td>Y(I,J)</td>
<td>Cartesian y coordinate or cylindrical r coordinate. (METRIC1)</td>
<td></td>
</tr>
<tr>
<td>YMAX</td>
<td>Maximum y coordinate for Cartesian grid option. (GMTRY1)</td>
<td></td>
</tr>
<tr>
<td>YMIN</td>
<td>Minimum y coordinate for Cartesian grid option. (GMTRY1)</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Common Variables Listed Symbolically

In this section many of the Proteus Fortran variables stored in common blocks are defined, listed symbolically. Note that this list does not include those variables without symbolic representations, such as various flags, or those whose meaning depends on other parameters, such as the boundary condition values and sweep direction metrics. The variables marked with an asterisk are input variables. More details on these may be found in Section 3.1 of Volume 2. The common block each variable is stored in is given in parentheses at the end of each definition. For subscripted variables, the subscripts are defined along with the variable, except for the subscripts II and I2, which are the indices $i$ and $j$ in the $\xi$ and $\eta$ directions, respectively, and run from 1 to $N_1$ and $N_2$.

Unless otherwise noted, all variables representing physical quantities are nondimensional. The nondimensionalizing procedure is described in Section 3.1.1 of Volume 2. The type (real or integer) of the variables follows standard Fortran convention, unless stated otherwise. (I.e., those starting with I, J, K, L, M, or N are integer, and the remainder are real.)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Fortran Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^*$</td>
<td>APLUS</td>
<td>Van Driest damping constant in the inner and outer regions of the Baldwin-Lomax turbulence model. (TURB1)</td>
</tr>
<tr>
<td>A</td>
<td>AMAT1(I)</td>
<td>Subdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. In actual use, this one-dimensional array is equivalenced to the four-dimensional array A(IV,IS,J,K). IS is the grid index in the sweep direction, running from 1 to $N$. IV is the grid index in the &quot;vectorized&quot; direction (i.e., the non-sweep direction in which the &quot;BLK&quot; routines are vectorized), and runs from 2 to $N - 1$. The subscript $J = 1$ to $N_q$, corresponding to the $N_q$ coupled governing equations, and $K = 1$ to $N_q$, corresponding to the $N_q$ dependent variables. (NUM1)</td>
</tr>
<tr>
<td>$B^*$</td>
<td>CB</td>
<td>Constant used in the formula for the Klebanoff intermittency factor $F_{\delta t}$ in the outer region of the Baldwin-Lomax turbulence model, and in the inner region of the Spalding-Kleinstein turbulence model. (TURB1)</td>
</tr>
<tr>
<td>B</td>
<td>BMAT1(I)</td>
<td>Diagonal submatrix of coefficients in the block tridiagonal coefficient matrix. In actual use, this one-dimensional array is equivalenced to the four-dimensional array B(IV,IS,J,K). IS is the grid index in the sweep direction, running from 1 to $N$. IV is the grid index in the &quot;vectorized&quot; direction (i.e., the non-sweep direction in which the &quot;BLK&quot; routines are vectorized), and runs from 2 to $N - 1$. The subscript $J = 1$ to $N_q$, corresponding to the $N_q$ coupled governing equations, and $K = 1$ to $N_q$, corresponding to the $N_q$ dependent variables. (NUM1)</td>
</tr>
<tr>
<td>$c_p$</td>
<td>CP(11,12)</td>
<td>Specific heat at constant pressure at time level $n$. (FLOW1)</td>
</tr>
<tr>
<td>$c_v$</td>
<td>CV(11,12)</td>
<td>Specific heat at constant volume at time level $n$. (FLOW1)</td>
</tr>
<tr>
<td>$C_p$</td>
<td>CCP</td>
<td>Constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)</td>
</tr>
</tbody>
</table>
\( C_{cp1} - C_{cp4} \)  \( \text{CCP1-CCP4} \)
Constants in empirical formula for specific heat as a function of temperature. (FLOW1)

\( C_{x1} - C_{x2} \)  \( \text{CK1-2} \)
Constants in empirical formula for thermal conductivity coefficient as a function of temperature.

* \( C_{KLb} \)  \( \text{CKLEB} \)
Constant used in the formula for the Klebanoff intermittency factor \( F_{Klb} \) in the outer region of the Baldwin-Lomax turbulence model. (TURB1)

* \((C_{KLb})_{\min}\)  \( \text{CKMIN} \)
Constant used in the formula for the Klebanoff intermittency factor \( F_{Klb} \) in the outer region of the Baldwin-Lomax turbulence model. (TURB1)

* \( C_{\alpha} \)  \( \text{CMUR} \)
Constant used to compute \( C_{\alpha} \) in the turbulent viscosity formula for the \( k-\epsilon \) equations. (TURB20)

\( C_{\mu1} - C_{\mu2} \)  \( \text{CMU1-2} \)
Constants in empirical formula for laminar viscosity coefficient as a function of temperature. (FLOW1)

* \( C_{wk} \)  \( \text{CWK} \)
Constant used in the formula for \( F_{wke} \) in the outer region of the Baldwin-Lomax turbulence model. (TURB1)

* \( C_{\tau} \)  \( \text{CONE} \)
Constant used in the production term of the \( \epsilon \) equation. (TURB20)

* \( C_{2r} \)  \( \text{CTWOR} \)
Constant used to compute \( C_{2r} \) in the dissipation term of the \( \epsilon \) equation. (TURB20)

* \( C_{3} \)  \( \text{CTHREE} \)
Constant used to compute \( C_{3} \) in the turbulent viscosity formula for the \( k-\epsilon \) equations. (TURB20)

\( \mathbf{C} \)  \( \text{CMAT1(I)} \)
Superdiagonal submatrix of coefficients in the block tridiagonal coefficient matrix. In actual use, this one-dimensional array is equivalenced to the four-dimensional array \( \mathbf{C}(IV,IS,JI,K) \). \( IS \) is the grid index in the sweep direction, running from 1 to \( N \). \( IV \) is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N-1 \). The subscript \( J = 1 \) to \( N_{eo} \), corresponding to the \( N_{eo} \) coupled governing equations, and \( K = 1 \) to \( N_{ev} \), corresponding to the \( N_{ev} \) dependent variables. (NUM1)

\( e_{r} \)  \( \text{ER} \)
Dimensional reference energy, \( \rho \cdot u^2 \). (FLOW1)

\( E_{T} \)  \( \text{ET(I1,I2)} \)
Total energy at time level \( n \). (FLOW1)

\( E_{\tau} \)  \( \text{ETL(I1,I2)} \)
Total energy at previous or intermediate time level. (FLOW1)

\( g_{e} \)  \( \text{GC} \)
Dimensional proportionality factor in Newton's second law, either 32.174 \( \text{lb}_{m} \cdot \text{ft}/\text{lb}_{r} \cdot \text{sec}^2 \), or 1.0 \( \text{kg} \cdot \text{m}/\text{N} \cdot \text{sec}^2 \). (FLOW1)

\( h_{T} \)  \( \text{HSTAG} \)
Stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)

* \( h_{fr} \)  \( \text{HSTAGR} \)
Dimensional stagnation enthalpy used with constant stagnation enthalpy option. (FLOW1)

\( i \)  \( \text{I1} \)
Grid point index in the \( \xi \) direction. (NUM1)
\begin{itemize}
  \item \(i, j\) \quad IV \quad \text{Grid point index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized). Therefore, } IV = j \text{ for the first sweep and } i \text{ for the second sweep. (NUM1)}
  \\
  \item \(j^{-1}\) \quad JI(11,12) \quad \text{Inverse Jacobian of the non-orthogonal grid transformation. (For axisymmetric flow, in the COEF routines } JI = rJ^{-1}, \text{ the product of the local radius and the inverse Jacobian.) This is a real variable. (METRIC1)}
  \\
  \item \(k\) \quad KT(11,12) \quad \text{Effective thermal conductivity coefficient at time level } n. \text{ This is a real variable. (FLOW1)}
  \\
  \item \(k\) \quad KE(11,12) \quad \text{Turbulent kinetic energy at time level } n. \text{ This is a real variable. (TURB20)}
  \\
  \item \(k\) \quad KEL(11,12) \quad \text{Turbulent kinetic energy at previous or intermediate time level. This is a real variable. (TURB20)}
  \\
  \item \(k\) \quad KTR \quad \text{Dimensional reference thermal conductivity coefficient. This is a real variable. (FLOW1)}
  \\
  \item \(K\) \quad CCLAU \quad \text{Clauser constant used in the outer region of the Baldwin-Lomax turbulence model. (TURB1)}
  \\
  \item \(L_r\) \quad LR \quad \text{Dimensional reference length. This is a real variable. (FLOW1)}
  \\
  \item \(M_r\) \quad MACHR \quad \text{Reference Mach number, } u_0/(\gamma R T_0)^{1/2}. \text{ This is a real variable. (FLOW1)}
  \\
  \item \(n\) \quad IT \quad \text{Current time step number, or known time level. Time step number } n \text{ updates the solution from time level } n \text{ to } n + 1. \text{ (TIME1)}
  \\
  \item \(n\) \quad CNA \quad \text{Exponent in the formula used to average the two outer region } \mu, \text{ profiles that result when both boundaries in a coordinate direction are solid surfaces. (TURB1)}
  \\
  \item \(n\) \quad CNL \quad \text{Exponent in the Launder-Priddin modified mixing length formula for the inner region of the Baldwin-Lomax turbulence model. (TURB1)}
  \\
  \item \(N\) \quad NPTS \quad \text{The number of grid points in the sweep direction. (NUM1)}
  \\
  \item \(N_{eq}\) \quad NEQ \quad \text{The number of coupled governing equations actually being solved. (NUM1)}
  \\
  \item \(N_v\) \quad NV \quad \text{The number of grid points in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized). Therefore, } NV = N_2 \text{ for the first sweep and } N_1 \text{ for the second sweep. (NUM1)}
  \\
  \item \(N_1\) \quad N1 \quad \text{The number of grid points in the } \xi \text{ direction. (NUM1)}
\end{itemize}
\( N_1 \) \( \text{NPT1} \) The number of grid points in the \( \xi \) direction used in computing coefficients (only for non-periodic boundary conditions.) \( \text{(NUM1)} \)

\( N_1 + 1 \) \( \text{NPT1} \) The number of grid points in the \( \xi \) direction used in computing coefficients (only for spatially periodic boundary conditions.) \( \text{(NUM1)} \)

\* \( N_2 \) \( \text{N2} \) The number of grid points in the \( \eta \) direction. \( \text{(NUM1)} \)

\( N_2 \) \( \text{NPT2} \) The number of grid points in the \( \eta \) direction used in computing coefficients (only for non-periodic boundary conditions.) \( \text{(NUM1)} \)

\( N_2 + 1 \) \( \text{NPT2} \) The number of grid points in the \( \eta \) direction used in computing coefficients (only for spatially periodic boundary conditions.) \( \text{(NUM1)} \)

\( p \) \( P(I_{11}, I_{12}) \) Static pressure at time level \( n \). \( \text{(FLOW1)} \)

\( p_i \) \( \text{PR} \) Dimensional reference static pressure, \( \rho_r R T_1 g_c \). \( \text{(FLOW1)} \)

\* \( p_0 \) \( \text{P0} \) Initial static pressure. \( \text{(IC1)} \)

\( \partial p / \partial E_T \) \( \text{DPDET}(I) \) The derivative of \( p \) with respect to \( E_T \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). \( \text{(FLOW1)} \)

\( \partial p / \partial \rho \) \( \text{DPDRHO}(I) \) The derivative of \( p \) with respect to \( \rho \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). \( \text{(FLOW1)} \)

\( \partial p / \partial (\rho u) \) \( \text{DPDRU}(I) \) The derivative of \( p \) with respect to \( \rho u \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). \( \text{(FLOW1)} \)

\( \partial p / \partial (\rho v) \) \( \text{DPDRV}(I) \) The derivative of \( p \) with respect to \( \rho v \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). \( \text{(FLOW1)} \)

\( \partial p / \partial (\rho w) \) \( \text{DPDRW}(I) \) The derivative of \( p \) with respect to \( \rho w \), stored as a one-dimensional array in the sweep direction. The subscript \( I \) therefore runs from 1 to \( N \). \( \text{(FLOW1)} \)

\( P_1 \) \( \text{PONE} \) Part 1 of the production rate of the turbulent kinetic energy. \( \text{(TURB20)} \)

\( P_2 \) \( \text{PTWO} \) Part 2 of the production rate of the turbulent kinetic energy. \( \text{(TURB20)} \)

\* \( Pr_r \) \( \text{PRLR} \) Reference laminar Prandtl number, \( c_p \mu / k_r \), where \( c_p = \gamma, R / (\gamma - 1) \). \( \text{(FLOW1)} \)

\( Pr_r \) \( \text{PRR} \) Reference Prandtl number, \( \mu a / k_r T_r \). \( \text{(FLOW1)} \)
* \( p_{rt} \) \hspace{1cm} \text{PRT} \hspace{1cm} \text{Turbulent Prandtl number, or, if non-positive, a flag indicating the use of a variable turbulent Prandtl number. (TURB1)}

\[ \Delta Q_{\text{avg}} \] \hspace{1cm} \text{CHGAVG(I)} \hspace{1cm} \text{Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables. (TIME1)}

\[ \Delta Q_{\text{max}} \] \hspace{1cm} \text{CHGMAX(I,J)} \hspace{1cm} \text{Maximum change in absolute value of the dependent variables over a single time step. The subscript I = 1 to N_{eq}, corresponding to the N_{eq} dependent variables, and J = 1 to NITAVG, the number of time steps used in the moving average option for determining convergence. (TIME1)}

\( r \) \hspace{1cm} Y(I1,I2) \hspace{1cm} \text{Cylindrical } r \text{ coordinate. (METRIC1)}

\( r \) \hspace{1cm} \text{RAX(I)} \hspace{1cm} \text{Local radius } r \text{ for axisymmetric flow. I is the grid index in the sweep direction, running from 1 to N. (METRIC1)}

* \( r'_{\text{max}} \) \hspace{1cm} \text{RMAX} \hspace{1cm} \text{Maximum } r' \text{ coordinate coordinate for polar grid option. (GMTRY1)}

* \( r'_{\text{min}} \) \hspace{1cm} \text{RMIN} \hspace{1cm} \text{Minimum } r' \text{ coordinate coordinate for polar grid option. (GMTRY1)}

\( R_{eq} \) \hspace{1cm} \text{RESAVG(J,K)} \hspace{1cm} \text{The average absolute value of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)}

\( R_{L2} \) \hspace{1cm} \text{RESL2(J,K)} \hspace{1cm} \text{The } L_2 \text{ norm of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)}

\( R_{max} \) \hspace{1cm} \text{RESMAX(J,K)} \hspace{1cm} \text{The maximum absolute value of the residual for the previous time step. The subscript J = 1 to N_{eq}, corresponding to the N_{eq} coupled governing equations, and K = 1 or 2, corresponding to the residual computed without and with the artificial viscosity terms. (TIME1)}

* \( \bar{R} \) \hspace{1cm} \text{RG} \hspace{1cm} \text{Dimensional gas constant. (FLOW1)}

\( R \) \hspace{1cm} \text{RGAS} \hspace{1cm} \text{Nondimensional gas constant. (FLOW1)}

* \( Re \) \hspace{1cm} \text{RER} \hspace{1cm} \text{Reference Reynolds number, } \rho u L / \mu . \text{ (FLOW1)}

* \( Re_{e,\text{tr}} \) \hspace{1cm} \text{REXT1} \hspace{1cm} \text{Reynolds number at the beginning of the transition region, based on maximum total velocity and distance from } \xi = 0, \text{ for flow predominantly in the } \xi \text{ direction with a leading edge at } \xi = 0. \text{ (TURB1)}

---

6 For the energy equation, the change in \( E_r \) is divided by \( E_r = \rho \bar{R} T_s (r, -1) + u^2 / 2 \), so that it is the same order of magnitude as the other conservation variables.
* $Re_r$ REXT2 Reynolds number at the beginning of the transition region, based on maximum total velocity and distance from $\eta = 0$, for flow predominantly in the $\eta$ direction with a leading edge at $\eta = 0$. (TURB1)

$S$ SVCT1(I) Subvector of source terms in the block tridiagonal system of equations. In actual use, this one-dimensional array is equivalenced to the three-dimensional array $S(IV,IS,J)$. IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the “vectorized” direction (i.e., the non-sweep direction in which the “BLK” routines are vectorized), and runs from 2 to $N_r - 1$. The subscript $J = 1$ to $N_q$, corresponding to the $N_q$ coupled governing equations. (NUM1)

* $\Delta t$ DT(I) The time step size, when specified directly as input. I is the time step sequence number, and runs from 1 to $NTSEQ$. (TIME1)

$T$ T(I1,I2) Static temperature at time level $n$. (FLOW1)

$T$ TL(I1,I2) Static temperature at previous or intermediate time level. (FLOW1)

$\partial T/\partial E_r$ DTDET(I) The derivative of $T$ with respect to $E_r$, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)

$\partial T/\partial \rho$ DTDRHO(I) The derivative of $T$ with respect to $\rho$, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)

$\partial T/\partial (\rho u)$ DTDRU(I) The derivative of $T$ with respect to $\rho u$, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)

$\partial T/\partial (\rho v)$ DTDRV(I) The derivative of $T$ with respect to $\rho v$, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)

$\partial T/\partial (\rho w)$ DTDRW(I) The derivative of $T$ with respect to $\rho w$, stored as a one-dimensional array in the sweep direction. The subscript I therefore runs from 1 to N. (FLOW1)

* $T_r$ TR Dimensional reference temperature. (FLOW1)

* $T_0$ T0 Initial static temperature. (IC1)

$u$ U(I1,I2) Velocity in the Cartesian $x$ direction at time level $n$. (FLOW1)

$u$ UL(I1,I2) Velocity in the Cartesian $x$ direction at previous or intermediate time level. (FLOW1)

* $u_r$ UR Dimensional reference velocity. (FLOW1)

* $u_0$ U0 Initial velocity in the Cartesian $x$ direction. (IC1)

$v$ V(I1,I2) Velocity in the Cartesian $y$ direction or cylindrical $r$ direction at time level $n$. (FLOW1)
v  \( VL(I1,I2) \)  Velocity in the Cartesian \( y \) direction or cylindrical \( r \) direction at previous or intermediate time level. (FLOW1)

* \( v_0 \)  V0 Initial velocity in the Cartesian \( y \) direction or cylindrical \( r \) direction. (IC1)

w  \( W(I1,I2) \)  Swirl velocity at time level \( n \). (FLOW1)

\( \omega \)  \( WL(I1,I2) \)  Swirl velocity at previous or intermediate time level. (FLOW1)

* \( \omega_0 \)  W0 Initial swirl velocity. (IC1)

x  \( X(I1,I2) \)  Cartesian \( x \) coordinate. (METRIC1)

* \( x_{\text{max}} \)  XMAX Maximum \( x \) coordinate for Cartesian grid option. (GMTRY1)

* \( x_{\text{min}} \)  XMIN Minimum \( x \) coordinate for Cartesian grid option. (GMTRY1)

y  \( Y(I1,I2) \)  Cartesian \( y \) coordinate. (METRIC1)

* \( y_{\text{max}} \)  YMAX Maximum \( y \) coordinate for Cartesian grid option. (GMTRY1)

* \( y_{\text{min}} \)  YMIN Minimum \( y \) coordinate for Cartesian grid option. (GMTRY1)

\( y^+ \)  \( YPLUSD(I1,I2) \)  Non-dimensional distance from the nearest solid wall. (TURB20)

* \( \varepsilon \)  \( EPS(I) \)  Convergence level to be reached. The subscript \( I = 1 \) to \( N_q \), corresponding to the \( N_q \) dependent variables. (TIME1)

\( \varepsilon \)  \( E(I1,I2) \)  Turbulent dissipation rate at time level \( n \). (TURB20)

\( \varepsilon \)  \( EL(I1,I2) \)  Turbulent dissipation rate at previous or intermediate time level. (TURB20)

* \( \epsilon_2 \)  \( CAVS2E(I) \)  Second-order explicit artificial viscosity coefficient in constant coefficient model. The subscript \( I = 1 \) to \( N_q \), corresponding to the \( N_q \) coupled governing equations. (NUM1)

* \( \epsilon_4 \)  \( CAVS4E(I) \)  Fourth-order explicit artificial viscosity coefficient in constant coefficient model. The subscript \( I = 1 \) to \( N_q \), corresponding to the \( N_q \) coupled governing equations. (NUM1)

* \( \epsilon_6 \)  \( CAVS2I(I) \)  Second-order implicit artificial viscosity coefficient in constant coefficient model. The subscript \( I = 1 \) to \( N_q \), corresponding to the \( N_q \) coupled governing equations. (NUM1)

\( \eta_r \)  \( ETAY(I1,I2) \)  The derivative of the computational coordinate \( \eta \) with respect to the cylindrical coordinate \( r \). (METRIC1)

\( \eta_r \)  \( METY1(I) \)  The derivative of the computational coordinate \( \eta \) with respect to the cylindrical coordinate \( r \) (second ADI sweep only.) In actual use, this one-dimensional array is equivalenced to the
two-dimensional array METY(IV,IS). IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N - 1 \). This is a real variable. (METRIC1)

\( \eta_t \) ETAT(I1,I2) The derivative of the computational coordinate \( \eta \) with respect to untransformed time \( t \). (METRIC1)

\( \eta_t \) METT1(I) The derivative of the computational coordinate \( \eta \) with respect to untransformed time \( t \) (second ADI sweep only.) In actual use, this one-dimensional array is equivalent to the two-dimensional array METT(IV,IS). IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N - 1 \). This is a real variable. (METRIC1)

\( \eta_x \) ETAX(I1,I2) The derivative of the computational coordinate \( \eta \) with respect to the Cartesian coordinate \( x \). (METRIC1)

\( \eta_x \) METX1(I) The derivative of the computational coordinate \( \eta \) with respect to the Cartesian coordinate \( x \) (second ADI sweep only.) In actual use, this one-dimensional array is equivalent to the two-dimensional array METX(IV,IS). IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N - 1 \). This is a real variable. (METRIC1)

\( \eta_y \) ETAY(I1,I2) The derivative of the computational coordinate \( \eta \) with respect to the Cartesian coordinate \( y \). (METRIC1)

\( \eta_y \) METY1(I) The derivative of the computational coordinate \( \eta \) with respect to the Cartesian coordinate \( y \) (second ADI sweep only.) In actual use, this one-dimensional array is equivalent to the two-dimensional array METY(IV,IS). IS is the grid index in the sweep direction, running from 1 to N. IV is the grid index in the "vectorized" direction (i.e., the non-sweep direction in which the "BLK" routines are vectorized), and runs from 2 to \( N - 1 \). This is a real variable. (METRIC1)

\( \Delta \eta \) DEL Computational grid spacing in the \( \eta \) direction (second ADI sweep only.) (NUM1)

\( \Delta \eta \) DETA Computational grid spacing in the \( \eta \) direction. (NUM1)

* \( \kappa \) CVK Von Karman mixing length constant used in the inner region of the Baldwin-Lomax and Spalding-Kleinstein turbulence models. (TURB1)

* \( \kappa_2 \) CAVS2E(I) User-specified constant in nonlinear coefficient artificial viscosity model. The subscript \( I = 1 \) to \( N_{eq} \), corresponding to the \( N_{eq} \) coupled governing equations. (NUM1)

* \( \kappa_4 \) CAVS4E(I) User-specified constant in nonlinear coefficient artificial viscosity model. The subscript \( I = 1 \) to \( N_{eq} \), corresponding to the \( N_{eq} \) coupled governing equations. (NUM1)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma_r$</td>
<td>Reference ratio of specific heats, $c_p/c_v$</td>
<td>(FLOW1)</td>
</tr>
<tr>
<td>$\chi$</td>
<td>Effective second coefficient of viscosity at time level $n$ (usually assumed equal to $-2\mu/3$.) This is a real variable.</td>
<td>(FLOW1)</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Effective viscosity coefficient at time level $n$. This is a real variable.</td>
<td>(FLOW1)</td>
</tr>
<tr>
<td>$\mu_r$</td>
<td>Dimensional reference viscosity coefficient. This is a real variable.</td>
<td>(FLOW1)</td>
</tr>
<tr>
<td>$\mu_t$</td>
<td>Turbulent viscosity coefficient at time level $n$. This is a real variable.</td>
<td>(FLOW1)</td>
</tr>
<tr>
<td>$\mu_u$</td>
<td>Turbulent viscosity coefficient at previous or intermediate time level. This is a real variable.</td>
<td>(TURB20)</td>
</tr>
<tr>
<td>$\xi_r$</td>
<td>The derivative of the computational coordinate $\xi$ with respect to the cylindrical coordinate $r$.</td>
<td>(METRIC1)</td>
</tr>
<tr>
<td>$\xi_t$</td>
<td>The derivative of the computational coordinate $\xi$ with respect to untransformed time $t$.</td>
<td>(METRIC1)</td>
</tr>
<tr>
<td>$\xi_u$</td>
<td>The derivative of the computational coordinate $\xi$ with respect to untransformed time $t$ (first ADI sweep only.) In actual use, this one-dimensional array is equivalenced to the two-dimensional array METT(IV,IS).</td>
<td>(METRIC1)</td>
</tr>
<tr>
<td>$\xi_x$</td>
<td>The derivative of the computational coordinate $\xi$ with respect to the Cartesian coordinate $x$.</td>
<td>(METRIC1)</td>
</tr>
<tr>
<td>$\xi_y$</td>
<td>The derivative of the computational coordinate $\xi$ with respect to the Cartesian coordinate $y$.</td>
<td>(METRIC1)</td>
</tr>
<tr>
<td>$\xi_{y_u}$</td>
<td>The derivative of the computational coordinate $\xi$ with respect to the Cartesian coordinate $y$ (first ADI sweep only.) In actual use, this one-dimensional array is equivalenced to the...</td>
<td>(METRIC1)</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>Δξ</td>
<td>Computational grid spacing in the ξ direction (first ADI sweep only.)</td>
<td>NUM1</td>
</tr>
<tr>
<td>Δξ</td>
<td>Computational grid spacing in the ξ direction.</td>
<td>NUM1</td>
</tr>
<tr>
<td>ρ</td>
<td>Static density at time level n.</td>
<td>FLOW1</td>
</tr>
<tr>
<td>ρ</td>
<td>Static density at previous or intermediate time level.</td>
<td>FLOW1</td>
</tr>
<tr>
<td>ρ_0</td>
<td>Dimensional reference density.</td>
<td>FLOW1</td>
</tr>
<tr>
<td>σ_k</td>
<td>Constant used in the diffusion term of the k equation.</td>
<td>TURB20</td>
</tr>
<tr>
<td>σ_e</td>
<td>Constant used in the diffusion term of the ε equation.</td>
<td>TURB20</td>
</tr>
<tr>
<td>τ</td>
<td>Current value of the time marching parameter.</td>
<td>TIME1</td>
</tr>
<tr>
<td>Δτ</td>
<td>Computational time step size.</td>
<td>TIME1</td>
</tr>
<tr>
<td>ϑ_{max}</td>
<td>Maximum ϑ coordinate in degrees for polar grid option.</td>
<td>GMTRY1</td>
</tr>
<tr>
<td>ϑ_{min}</td>
<td>Minimum ϑ coordinate in degrees for polar grid option.</td>
<td>GMTRY1</td>
</tr>
<tr>
<td>ϑ_1, ϑ_2</td>
<td>A two-element array specifying the time difference centering parameters used for the continuity equation.</td>
<td>NUM1</td>
</tr>
<tr>
<td>ϑ_1, ϑ_2</td>
<td>A two-element array specifying the time difference centering parameters used for the k-ε equations.</td>
<td>NUM2</td>
</tr>
<tr>
<td>ϑ_1, ϑ_2, ϑ_3</td>
<td>A three-element array specifying the time difference centering parameters used for the energy equation.</td>
<td>NUM1</td>
</tr>
<tr>
<td>ϑ_1, ϑ_2, ϑ_3</td>
<td>A three-element array specifying the time difference centering parameters used for the x-momentum equation.</td>
<td>NUM1</td>
</tr>
<tr>
<td>ϑ_1, ϑ_2, ϑ_3</td>
<td>A three-element array specifying the time difference centering parameters used for the y or r-momentum equation.</td>
<td>NUM1</td>
</tr>
<tr>
<td>ϑ_1, ϑ_2, ϑ_3</td>
<td>A three-element array specifying the time difference centering parameters used for the swirl momentum equation.</td>
<td>NUM1</td>
</tr>
<tr>
<td></td>
<td>Total vorticity magnitude.</td>
<td>TURB1</td>
</tr>
<tr>
<td></td>
<td>Production rate of turbulent kinetic energy.</td>
<td>TURB1</td>
</tr>
</tbody>
</table>
4.0 PROTEUS SUBPROGRAMS

In this section, each subprogram in Proteus is described, first in summary, then in detail. The summary is simply a list of the subprograms with a brief description of the purpose of each one. The detailed description includes, for each subprogram, a list of the subprograms that reference it, and a list of the subprograms that it references. The Fortran variables that are input to and output from each subprogram are defined. And finally, details of the computations being done within each subprogram are presented.

4.1 SUBPROGRAM SUMMARY

The following table presents a brief description of the purpose of each subprogram in the Proteus code.

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td>Manage the block tridiagonal inversion.</td>
</tr>
<tr>
<td>AVISC1</td>
<td>Compute constant coefficient artificial viscosity.</td>
</tr>
<tr>
<td>AVISC2</td>
<td>Compute nonlinear coefficient artificial viscosity.</td>
</tr>
<tr>
<td>BCDENS</td>
<td>Compute density boundary conditions.</td>
</tr>
<tr>
<td>BCELIM</td>
<td>Eliminate off-diagonal coefficient submatrices resulting from three-point boundary conditions.</td>
</tr>
<tr>
<td>BCF</td>
<td>Compute user-written boundary conditions.</td>
</tr>
<tr>
<td>BCFLIN</td>
<td>User-supplied routine for linearization of user-supplied boundary conditions.</td>
</tr>
<tr>
<td>BCGEN</td>
<td>Manage computation of boundary conditions.</td>
</tr>
<tr>
<td>BCGRAD</td>
<td>Compute gradients with respect to $\xi$ and $\eta$.</td>
</tr>
<tr>
<td>BCMET</td>
<td>Compute various metric functions for normal gradient boundary conditions.</td>
</tr>
<tr>
<td>BCPRES</td>
<td>Compute pressure boundary conditions.</td>
</tr>
<tr>
<td>BCQ</td>
<td>Compute conservation variable boundary conditions.</td>
</tr>
<tr>
<td>BCSET</td>
<td>Set various boundary condition parameters and flags.</td>
</tr>
<tr>
<td>BCTEMP</td>
<td>Compute temperature boundary conditions.</td>
</tr>
<tr>
<td>BCUVEL</td>
<td>Compute $x$-velocity boundary conditions.</td>
</tr>
<tr>
<td>BCVDIR</td>
<td>Compute normal and tangential velocity boundary conditions.</td>
</tr>
<tr>
<td>BCSVEL</td>
<td>Compute $y$ or $r$-velocity boundary conditions.</td>
</tr>
<tr>
<td>BCWVEL</td>
<td>Compute swirl velocity boundary conditions.</td>
</tr>
<tr>
<td>BLIN1</td>
<td>Compute inner layer turbulent viscosity along constant $\xi$ lines.</td>
</tr>
<tr>
<td>BLIN2</td>
<td>Compute inner layer turbulent viscosity along constant $\eta$ lines.</td>
</tr>
<tr>
<td>BLKOUT</td>
<td>Print coefficient blocks at specified indices in the $\xi$ and $\eta$ directions.</td>
</tr>
<tr>
<td>BLK2</td>
<td>Solve $2 \times 2$ block tridiagonal system of equations.</td>
</tr>
<tr>
<td>BLK2P</td>
<td>Solve $2 \times 2$ periodic block tridiagonal system of equations.</td>
</tr>
<tr>
<td>BLK3</td>
<td>Solve $3 \times 3$ block tridiagonal system of equations.</td>
</tr>
<tr>
<td>BLK3P</td>
<td>Solve $3 \times 3$ periodic block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>
# Proteus Subprogram Summary

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLK4</td>
<td>Solve $4 \times 4$ block tridiagonal system of equations.</td>
</tr>
<tr>
<td>BLK4P</td>
<td>Solve $4 \times 4$ periodic block tridiagonal system of equations.</td>
</tr>
<tr>
<td>BLK5</td>
<td>Solve $5 \times 5$ block tridiagonal system of equations.</td>
</tr>
<tr>
<td>BLK5P</td>
<td>Solve $5 \times 5$ periodic block tridiagonal system of equations.</td>
</tr>
<tr>
<td>BLOCK DATA</td>
<td>Set default values for input parameters, plus a few other parameters.</td>
</tr>
<tr>
<td>BLOUT1</td>
<td>Compute outer layer turbulent viscosity along constant $\xi$ lines.</td>
</tr>
<tr>
<td>BLOUT2</td>
<td>Compute outer layer turbulent viscosity along constant $\eta$ lines.</td>
</tr>
<tr>
<td>BVUP</td>
<td>Update first sweep boundary values after second sweep.</td>
</tr>
<tr>
<td>COEFC</td>
<td>Compute coefficients and source terms for the continuity equation.</td>
</tr>
<tr>
<td>COEFE</td>
<td>Compute coefficients and source terms for the energy equation.</td>
</tr>
<tr>
<td>COEFS1</td>
<td>Compute coefficients and source terms for the $k$ and $\varepsilon$ equations for the first ADI sweep.</td>
</tr>
<tr>
<td>COEFS2</td>
<td>Compute coefficients and source terms for the $k$ and $\varepsilon$ equations for the second ADI sweep.</td>
</tr>
<tr>
<td>COEFX</td>
<td>Compute coefficients and source terms for the $x$-momentum equation.</td>
</tr>
<tr>
<td>COEFY</td>
<td>Compute coefficients and source terms for the $y$ or $r$-momentum equation.</td>
</tr>
<tr>
<td>COEFZ</td>
<td>Compute coefficients and source terms for the swirl momentum equation.</td>
</tr>
<tr>
<td>CONV</td>
<td>Test computed flow field for convergence.</td>
</tr>
<tr>
<td>CUBIC</td>
<td>Interpolation using Ferguson's parametric cubic.</td>
</tr>
<tr>
<td>EQSTAT</td>
<td>Use equation of state to compute pressure, temperature, and their derivatives with respect to the dependent variables.</td>
</tr>
<tr>
<td>EXEC</td>
<td>Manage solution of the mean flow equations.</td>
</tr>
<tr>
<td>EXECT</td>
<td>Manage solution of the $k$-$\varepsilon$ equations.</td>
</tr>
<tr>
<td>FILTER</td>
<td>Rearrange rows of the boundary condition coefficient submatrices and the source term subvector to eliminate any zeroes on the diagonal.</td>
</tr>
<tr>
<td>FTEMP</td>
<td>Compute auxiliary variables that are functions of temperature.</td>
</tr>
<tr>
<td>GEOM</td>
<td>Manage computation of grid and metric parameters.</td>
</tr>
<tr>
<td>INIT</td>
<td>Get user-defined initial flow field.</td>
</tr>
<tr>
<td>INITC</td>
<td>Set up consistent initial conditions based on data from INIT.</td>
</tr>
<tr>
<td>INPUT</td>
<td>Read and print input, perform various initializations.</td>
</tr>
<tr>
<td>ISAMAX</td>
<td>Find the first index corresponding to the largest absolute value of the elements of an vector. This is a Cray search routine.</td>
</tr>
<tr>
<td>ISAMIN</td>
<td>Find the first index corresponding to the smallest absolute value of the elements of an vector. This is a Cray search routine.</td>
</tr>
<tr>
<td>ISRCHEQ</td>
<td>Find the first index in an array whose element is equal to a specified value. This is a Cray search routine.</td>
</tr>
<tr>
<td>KEINIT</td>
<td>Get user-defined initial conditions for $k$ and $\varepsilon$.</td>
</tr>
<tr>
<td>MAIN</td>
<td>Manage overall solution.</td>
</tr>
<tr>
<td>METS</td>
<td>Compute metrics of nonorthogonal grid transformation.</td>
</tr>
</tbody>
</table>
## Proteus Subprogram Summary

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>Manage printing of output.</td>
</tr>
<tr>
<td>OUTW</td>
<td>Compute and print parameters at boundaries.</td>
</tr>
<tr>
<td>PAK</td>
<td>Manage packing and/or interpolation of grid points.</td>
</tr>
<tr>
<td>PERIOD</td>
<td>Define extra line of data for use in computing coefficients for spatially periodic boundary conditions.</td>
</tr>
<tr>
<td>PLOT</td>
<td>Write files for post-processing by CONTOUR or PLOT3D plotting programs.</td>
</tr>
<tr>
<td>PRODCT</td>
<td>Compute production term for the k-ε turbulence model.</td>
</tr>
<tr>
<td>PRTOUT</td>
<td>Print output.</td>
</tr>
<tr>
<td>RESID</td>
<td>Compute residuals and write convergence history file.</td>
</tr>
<tr>
<td>ROBTS</td>
<td>Pack points along a line using Roberts transformation.</td>
</tr>
<tr>
<td>SASUM</td>
<td>Compute the sum of the absolute values of the elements of a vector. This is a Cray BLAS routine.</td>
</tr>
<tr>
<td>SGEFA</td>
<td>Factor a matrix using Gaussian elimination. This is a Cray LINPACK routine.</td>
</tr>
<tr>
<td>SGESL</td>
<td>Solve the matrix equation ( Ax = B ) or ( A^T x = B ) using the factors computed by SGEFA. This is a Cray LINPACK routine.</td>
</tr>
<tr>
<td>SNRM2</td>
<td>Compute the ( L_2 ) norm of a vector. This is a Cray BLAS routine.</td>
</tr>
<tr>
<td>TBC</td>
<td>Set time-dependent boundary condition values.</td>
</tr>
<tr>
<td>TIMSTP</td>
<td>Set computational time step.</td>
</tr>
<tr>
<td>TREMAIN</td>
<td>Get CPU time remaining for the job. This is a Cray Fortran routine.</td>
</tr>
<tr>
<td>TURBBL</td>
<td>Manage computation of turbulence parameters using the Baldwin-Lomax algebraic model.</td>
</tr>
<tr>
<td>TURBCH</td>
<td>Manage computation of turbulence parameters using the Chien k-ε model.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Update flow variables after each ADI sweep.</td>
</tr>
<tr>
<td>UPDTKE</td>
<td>Update ( k ) and ( \varepsilon ) after each time step.</td>
</tr>
<tr>
<td>VORTEX</td>
<td>Compute magnitude of total vorticity.</td>
</tr>
<tr>
<td>YPLUSN</td>
<td>Compute the distance to the nearest solid wall.</td>
</tr>
</tbody>
</table>

### 4.2 SUBPROGRAM DETAILS

The subprograms used in Proteus are described in detail in the remainder of this section. A few additional words are necessary about the input and output descriptions. The description of the input to each subprogram includes all Fortran variables actually used by the subprogram that are defined outside the subprogram. Variables defined and used inside the subprogram are not listed as input. In addition, common block variables that are merely passed through to lower level routines are not listed. Variables marked with an asterisk are user-specified namelist input variables.

Similarly, the output description includes only those variables computed inside the subprogram and used outside the subprogram. It does not include common block variables computed by lower level routines. In general, variables defined inside the subprogram that are used by lower level routines are listed as output, even if they are not needed after control is returned to the calling program.
Variables entering or leaving a subprogram through an argument list are defined in detail. However, most of the Fortran variables listed in the input and output descriptions are contained in common blocks, and are defined in detail in Section 3.0. For that reason, they are defined only briefly in this section.
Subroutine ADI (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>BLKOUT</td>
<td>Manage the block tridiagonal inversion.</td>
</tr>
<tr>
<td></td>
<td>BLK3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLK3P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLK4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLK4P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLK5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLK5P</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- A, B, C  
  Coefficient submatrices A, B, and C.
- IDEBUG   
  Debug flags.
- IPRT1A, IPRT2A  
  Indices for printout in the $\xi$ and $\eta$ directions.
- ISWEEP   
  Current ADI sweep number.
- IT       
  Current time step number $n$.
- KBCPER   
  Flags for spatially periodic boundary conditions in the $\xi$ and $\eta$ directions; 0 for non-periodic, 1 for periodic.
- NEQ      
  Number of coupled equations being solved, $N_{eq}$.
- NOUT     
  Unit number for standard output.
- NPRT1, NPRT2 
  Total number of indices for printout in the $\xi$ and $\eta$ directions.
- NVD, NPTSD  
  Leading two dimensions for the arrays A, B, C, and S.
- S        
  Source term subvector S, and computed solution subvector, $\Delta Q^+$ or $\Delta Q^{-}$.

**Output**

None.

**Description**

For each ADI sweep, subroutine ADI calls the appropriate block solver. The choice is determined by the number of equations being solved, and by the presence or absence of spatially periodic boundary conditions in the sweep direction.

**Remarks**

1. This subroutine generates the output for the IDEBUG(1), IDEBUG(5), and IDEBUG(6) options.
Subroutine AVISC1 (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>BLKOUT</td>
<td>Compute constant coefficient artificial viscosity.</td>
</tr>
</tbody>
</table>

**Input**

- A, B, C
- CAVS2E, CAVS4E, CAVS2I
- DTAU
- IAV2E, IAV4E, IAV2I
- IDEBUG
- IHSTAG
- IPRT1A, IPRT2A
- ISWEEP
- ISWIRL
- IT
- JI
- NC, NXM, NYM, NZM, NEN
- NOUT
- NPRT1, NPRT2
- NPT1, NPT2
- NR, NRU, NRV, NRW, NET
- NVD, NPTSD
- RHO, U, V, W, ET
- S

**Output**

- A, B, C
- S

**Description**

Subroutine AVISC1 adds explicit and/or implicit artificial viscosity to the governing equations, using the constant coefficient model of Steger (1978), as presented by Pulliam (1986b). The model is described in Section 8.1 of Volume 1. The explicit artificial viscosity may be second and/or fourth order, and is added only during the first ADI sweep. The implicit artificial viscosity is second order, and is added during both sweeps.
The fourth-order explicit artificial viscosity is implemented in Fortran by redefining the source term subvector as

\[ S_{i,j} = S_{i,j} - \frac{\epsilon_{E}^{(4)} \Delta \tau_{i,j}}{J_{i,j}} \left[ (\nabla_{x} \Delta)_{i,j}^{2} Q_{i,j} + (\nabla_{\eta} \Delta_{\eta})^{2} Q_{i,j} \right] \]

where \( i \) and \( j \) vary from 3 to \( NPT1 - 2 \) and from 3 to \( NPT2 - 2 \), respectively. At grid points adjacent to boundaries the fourth-order differences in the above equation cannot be used, and therefore are replaced by second-order differences. Thus, at \( i = 2 \) and at \( i = NPT1 - 1 \), with \( j \) varying from 3 to \( NPT2 - 2 \),

\[ S_{i,j} = S_{i,j} + \frac{\epsilon_{E}^{(4)} \Delta \tau_{i,j}}{J_{i,j}} \left[ (\nabla_{x} \Delta)_{i,j}^{2} Q_{i,j} - (\nabla_{\eta} \Delta_{\eta})^{2} Q_{i,j} \right] \]

Similarly, at \( j = 2 \) and at \( j = NPT2 - 1 \), with \( i \) varying from 3 to \( NPT1 - 1 \),

\[ S_{i,j} = S_{i,j} + \frac{\epsilon_{E}^{(4)} \Delta \tau_{i,j}}{J_{i,j}} \left[ - (\nabla_{x} \Delta)_{i,j}^{2} Q_{i,j} + \nabla_{\eta} \Delta_{\eta} Q_{i,j} \right] \]

The second-order explicit artificial viscosity is implemented in Fortran by redefining the source term subvector as

\[ S_{i,j} = S_{i,j} + \frac{\epsilon_{E}^{(2)} \Delta \tau_{i,j}}{J_{i,j}} \left[ (\nabla_{x} \Delta)_{i,j}^{2} Q_{i,j} + \nabla_{\eta} \Delta_{\eta} Q_{i,j} \right] \]

where \( i \) and \( j \) vary from 2 to \( NPT1 - 1 \) and from 2 to \( NPT2 - 1 \), respectively.

The second-order implicit artificial viscosity for the first ADI sweep is implemented in Fortran by redefining the coefficient block submatrices as

\[ A_{i,j} = A_{i,j} - \frac{\epsilon_{I} \Delta \tau_{i,j}}{J_{i,j}} J_{i-1,j} \]

\[ B_{i,j} = B_{i,j} + 2 \frac{\epsilon_{I} \Delta \tau_{i,j}}{J_{i,j}} J_{i,j} \]

\[ C_{i,j} = C_{i,j} - \frac{\epsilon_{I} \Delta \tau_{i,j}}{J_{i,j}} J_{i+1,j} \]

where \( i \) and \( j \) vary from 2 to \( NPT1 - 1 \) and from 2 to \( NPT2 - 1 \), respectively. Similarly, for the second sweep,

\[ A_{i,j} = A_{i,j} - \frac{\epsilon_{I} \Delta \tau_{i,j}}{J_{i,j}} J_{i,j-1} \]

\[ B_{i,j} = B_{i,j} + 2 \frac{\epsilon_{I} \Delta \tau_{i,j}}{J_{i,j}} J_{i,j} \]

\[ C_{i,j} = C_{i,j} - \frac{\epsilon_{I} \Delta \tau_{i,j}}{J_{i,j}} J_{i,j+1} \]

Remarks

1. The sign in front of each artificial viscosity term depends on the sign of the "\( i,j \)" term in the difference formula. See Section 8.1 of Volume 1 for details.
2. The coding to add artificial viscosity to the energy and/or swirl momentum equations is separate from the coding for the remaining equations, and is bypassed if they are not being solved.

3. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. For the first sweep (which includes all the explicit artificial viscosity) the order is thus (I2, I1), and for the second sweep the order is (I1, I2).

4. For spatially periodic boundary conditions in the $\xi$ direction, fourth-order differences could be used at $i = 2$ and at $i = NPT1 - 1 = N1$. Similarly, for the $\eta$ direction, fourth-order differences could be used at $j = 2$ and at $j = NPT2 - 1 = N2$. The logic to do this has not been coded, however, and at these points second-order differences are still used, as described above.

5. This subroutine generates the output for the IDEBUG(2) option.
Subroutine AVISC2 (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>BLKOUT</td>
<td>Compute nonlinear coefficient artificial viscosity.</td>
</tr>
</tbody>
</table>

**Input**

- **A, B, C**: Coefficient submatrices A, B, and C.
- **CAVS2E, CAVS4E**: User-specified coefficients $\kappa_2$ and $\kappa_4$.
- **CP, CV**: Specific heats $c_p$ and $c_v$ at time level $n$.
- **DTAU**: Time step $\Delta t$.
- **DXI, DETA**: Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **ETAX, ETAY, ETAT**: Metric coefficients $\eta_x$, $\eta_y$ (or $\eta_r$ if axisymmetric), and $\eta_t$.
- **IAV2E, IAV4E**: Flags for second-order and fourth-order explicit artificial viscosity.
- **IDEBUG**: Debug flags.
- **IHSTAG**: Flag for constant stagnation enthalpy option.
- **IPRT1A, IPRT2A**: Indices for printout in the $\xi$ and $\eta$ directions.
- **ISWEEP**: Current ADI sweep number.
- **ISWIRL**: Flag for swirl in axisymmetric flow.
- **IT**: Current time step number $n$.
- **JI**: Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.
- **NC, NXM, NYM, NZM, NEN**: Array indices associated with the continuity, $x$-momentum, $y$-momentum (or $r$-momentum if axisymmetric), swirl momentum, and energy equations.
- **NOUT**: Unit number for standard output.
- **NPRT1, NPRT2**: Total number of indices for printout in the $\xi$ and $\eta$ directions.
- **NPT1, NPT2**: $N_1$ and $N_2$ for non-periodic boundary conditions, $N_1 + 1$ and $N_2 + 1$ for spatially periodic boundary conditions in $\xi$ and $\eta$.
- **NVD, NPTSD**: Leading two dimensions for the arrays A, B, C, and S.
- **P, T**: Static pressure $p$ and temperature $T$ at time level $n$.
- **RGAS**: Gas constant $R$.
- **RHO, U, V, W, ET**: Static density $\rho$, velocities $u, v, w$, and total energy $E_T$ at time level $n$.
- **S**: Source term subvector S without artificial viscosity.
- **XIX, XIY, XIT**: Metric coefficients $\xi_x$, $\xi_y$ (or $\xi_r$ if axisymmetric), and $\xi_t$.

**Output**

- **S**: Source term subvector S with artificial viscosity.

**Description**

Subroutine AVISC2 adds explicit artificial viscosity to the governing equations, using the nonlinear coefficient model of Jameson, Schmidt, and Turkel (1981), as presented by Pulliam (1986b). The model is described in Section 8.2 of Volume 1. Implicit artificial viscosity is not normally used in combination with Proteus 2-D Programmer's Reference 4.0 Proteus Subprograms: AVISC2 57
the explicit nonlinear coefficient model. The explicit artificial viscosity is added only during the first ADI sweep.

The artificial viscosity in the $\xi$ direction is computed first, at the $\eta$-indices $j = 2$ to $NPT2 - 1$. The spectral radius term $\psi_{i,j}$ and the pressure gradient scaling factor $\sigma_{i,j}$ are computed and stored in local one-dimensional arrays for $i = 1$ to $NPT1$. Special formulas are used to compute $\sigma$ near boundaries, as described in Section 8.2 of Volume 1.

The second-order artificial viscosity is added first, and is implemented in Fortran by redefining the source term subvector as

$$S_{i,j} = S_{i,j} + \nabla_{\xi}' \left\{ \left( \frac{\psi}{J} \right)_{i+1,j} \left( \frac{\psi}{J} \right)_{i,j} \right\} (\epsilon^{(2)}_{\xi})_{i,j} \Delta_{\xi} Q_{i,j}$$

Or, after evaluating the differences,

$$S_{i,j} = S_{i,j} + \left[ \left( \frac{\psi}{J} \right)_{i+1,j} + \left( \frac{\psi}{J} \right)_{i,j} \right] (\epsilon^{(2)}_{\xi})_{i,j} (Q_{i+1,j} - Q_{i,j})$$

$$- \left[ \left( \frac{\psi}{J} \right)_{i,j} + \left( \frac{\psi}{J} \right)_{i-1,j} \right] (\epsilon^{(2)}_{\xi})_{i-1,j} (Q_{i,j} - Q_{i-1,j})$$

where $i$ varies from 2 to $NPT1 - 1$.

The fourth-order artificial viscosity is added next, and is implemented similarly by redefining the source term subvector as

$$S_{i,j} = S_{i,j} - \nabla_{\xi}' \left\{ \left( \frac{\psi}{J} \right)_{i+1,j} \left( \frac{\psi}{J} \right)_{i,j} \right\} (\epsilon^{(4)}_{\xi})_{i,j} \Delta_{\xi} \Delta_{\xi} Q_{i,j}$$

Or, after evaluating the differences,

$$S_{i,j} = S_{i,j} - \left[ \left( \frac{\psi}{J} \right)_{i+1,j} + \left( \frac{\psi}{J} \right)_{i,j} \right] (\epsilon^{(4)}_{\xi})_{i,j} (Q_{i+2,j} - 3Q_{i+1,j} + 3Q_{i,j} - Q_{i-1,j})$$

$$+ \left[ \left( \frac{\psi}{J} \right)_{i,j} + \left( \frac{\psi}{J} \right)_{i-1,j} \right] (\epsilon^{(4)}_{\xi})_{i-1,j} (Q_{i+1,j} - 3Q_{i,j} + 3Q_{i-1,j} - Q_{i-2,j})$$

where $i$ varies from 3 to $NPT1 - 2$. Special formulas are used at $i = 2$ and at $i = NPT1 - 1$, as described in Section 8.2 of Volume 1.

The artificial viscosity in the $\eta$ direction is computed next, and is implemented in a manner analogous to that just described for the artificial viscosity in the $\xi$ direction.

**Remarks**

1. The sign in front of each artificial viscosity term depends on the sign of the "$i,j$" term in the difference formula. See Section 8.1 of Volume 1 for details.

2. The coding to add artificial viscosity to the energy and/or swirl momentum equations is separate from the coding for the remaining equations, and is bypassed if they are not being solved.

3. The subscripts on the Fortran variable $S$ may be confusing. The first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. For the first sweep (which includes all the explicit artificial viscosity) the order is thus $(12,11)$.  

58 4.0 Proteus Subprograms: AVISC2 Proteus 2-D Programmer's Reference
4. For spatially periodic boundary conditions, the need for special formulas near boundaries could be eliminated. The logic to do this has not been coded, however.

5. This subroutine generates the output for the IDEBUG(2) option.
Subroutine BCDENS (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCGRAD BCMET</td>
<td>Compute density boundary conditions.</td>
</tr>
</tbody>
</table>

**Input**

- **DEL**
  Computational grid spacing in sweep direction.
- **IBASE, ISTEP**
  Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IBC, FBC**
  Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to \(N_q\), corresponding to the \(N_q\) conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
- **IBOUND**
  Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
- **IEQ**
  Boundary condition equation number.
- **IMIN, IMAX**
  Minimum and maximum indices in the sweep direction.
- **ISWEEP**
  Current ADI sweep number.
- **IV**
  Index in the "vectorized" direction, \(i\).
- **JI**
  Inverse Jacobian of the nonorthogonal grid transformation, \(J^{-1}\).
- **NOUT**
  Unit number for standard output.
- **NR**
  Array index associated with the dependent variable \(\rho\).
- **NVD, NPTSD**
  Leading two dimensions for the arrays A, B, C, and S.
- **RHO**
  Static density \(\rho\) at time level \(n\).

**Output**

- **A, B, C**
  Coefficient submatrices A, B, and C at boundary IBOUND (row IEQ only).
- **S**
  Source term subvector S at boundary IBOUND (element IEQ only).

**Description**

Subroutine BCDENS computes coefficients and source terms for density boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular density boundary conditions in *Proteus*.7

---

7 In the following description, for the first ADI sweep the dependent variable should have the superscript \(^*\), representing the intermediate solution, and for the second ADI sweep it should have the superscript \(n\), representing the final solution. For simplicity, however, only the superscript \(n\) is used. The superscripts on all other variables are correct as written.
No Change From Initial Conditions, $\Delta \rho = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial Q = J \partial g/\partial Q$, we get simply

$$J_{i,j} \Delta \hat{\rho}^n_{i,j} = 0$$

**Specified Static Density, $\rho = f$**

Applying equation (6.5) of Volume 1,

$$J_{i,j} \Delta \hat{\rho}^n_{i,j} = f_{i,j}^{n+1} - \rho_{i,j}^n$$

**Specified Two-Point Density Gradient in Coordinate Direction, $\partial \rho / \partial \phi = f$**

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j} \Delta \hat{\rho}^n_{1,j} + J_{2,j} \Delta \hat{\rho}^n_{2,j} = (\Delta \xi) f_{1,j}^{n+1} + \rho_{1,j}^n - \rho_{2,j}^n$$

At the $\xi = 1$ boundary,

$$-J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} + J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} = (\Delta \xi) f_{N_1,j}^{n+1} + \rho_{N_1,j}^n - \rho_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

**Specified Three-Point Density Gradient in Coordinate Direction, $\partial \rho / \partial \phi = f$**

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$-3J_{1,j} \Delta \hat{\rho}^n_{1,j} + 4J_{2,j} \Delta \hat{\rho}^n_{2,j} - J_{3,j} \Delta \hat{\rho}^n_{3,j} = 2(\Delta \xi) f_{1,j}^{n+1} + 3\rho_{1,j}^n - 4\rho_{2,j}^n + \rho_{3,j}^n$$

At the $\xi = 1$ boundary,

$$J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} - 2J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} - 4J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} + 3J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} = 2(\Delta \xi) f_{N_1,j}^{n+1} - \rho_{N_1,j}^n - 2\rho_{N_1,j} + 4\rho_{N_1,j} - 3\rho_{N_1,j}$$

Analogous equations can easily be written for the $\eta$ boundaries.

**Specified Two-Point Density Gradient in Normal Direction, $\nabla \rho \cdot \hat{n} = f$**

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j} \Delta \hat{\rho}^n_{1,j} + J_{2,j} \Delta \hat{\rho}^n_{2,j} = \frac{\Delta \xi}{m_{n,j}} \left[ f_{1,j}^{n+1} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{1,j}}{m_{n,j}} \delta \rho_{1,j}^n \right] + \rho_{1,j}^n - \rho_{2,j}^n$$

where

$$m = \sqrt{\xi_x^2 + \xi_y^2}$$

and $\delta$ is the centered difference operator presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$-J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} - 1,j + J_{N_1,j} \Delta \hat{\rho}^n_{N_1,j} = \frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{N_1,j}}{m_{N_1,j}} \delta \rho_{N_1,j}^n \right] + \rho_{N_1,j}^n - 1,j - \rho_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.
Specified Three-Point Density Gradient in Normal Direction, \( \nabla \rho \cdot \hat{n} = f \)

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[
-3J_{1,j} \Delta \rho_{1,j}^n + 4J_{2,j} \Delta \rho_{2,j}^n - J_{3,j} \Delta \rho_{3,j}^n = \frac{2\Delta \xi}{m_{1,j}} \left[ f_{1,j}^n + 1 - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{1,j}}{m_{1,j}} \delta_n \rho_{1,j}^n \right] + 3\rho_{1,j}^n - 4\rho_{2,j}^n + \rho_{3,j}^n
\]

where

\[
m = \sqrt{\xi_x^2 + \xi_y^2}
\]

and \( \delta_n \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
J_{N_1 - 2,j} \Delta \rho_{N_1 - 2,j}^n - 2J_{N_1 - 1,j} \Delta \rho_{N_1 - 1,j}^n + 3J_{N_1,j} \Delta \rho_{N_1,j}^n = \frac{2\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^n + 1 - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{N_1,j}}{m_{N_1,j}} \delta_n \rho_{N_1,j}^n \right] - \rho_{N_1 - 2,j}^n + 4\rho_{N_1 - 1,j}^n - 3\rho_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

Linear Extrapolation of Static Density

Applying equation (6.14) of Volume 1 at the \( \xi = 0 \) boundary,

\[
J_{1,j} \Delta \rho_{1,j}^n - 2J_{2,j} \Delta \rho_{2,j}^n + J_{3,j} \Delta \rho_{3,j}^n = - \rho_{1,j}^n + 2\rho_{2,j}^n - \rho_{3,j}^n
\]

At the \( \xi = 1 \) boundary,

\[
J_{N_1 - 2,j} \Delta \rho_{N_1 - 2,j}^n - 2J_{N_1 - 1,j} \Delta \rho_{N_1 - 1,j}^n + J_{N_1,j} \Delta \rho_{N_1,j}^n = - \rho_{N_1 - 2,j}^n + 2\rho_{N_1 - 1,j}^n - \rho_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent density boundary condition is specified.
Subroutine BCELIM (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>SGEFA</td>
<td>Eliminate off-diagonal coefficient submatrices resulting from three-point boundary conditions.</td>
</tr>
<tr>
<td></td>
<td>SGESL</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

A, B, C  
Coefficient submatrices A, B, and C before eliminating off-diagonal blocks.

IBCELM  
Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the \( \xi \) and \( \eta \) directions at either boundary; 0 if elimination is not necessary, 1 if it is.

ISWEEP  
Current ADI sweep number.

IV  
Index in the "vectorized" direction, \( i \).

NEQ  
Number of coupled equations being solved, \( N_q \).

NEQP  
Dimensioning parameter specifying maximum number of coupled equations allowed.

NPTS  
Number of grid points in the sweep direction, \( N \).

NVD, NPTSD  
Leading two dimensions for the arrays A, B, C, and S.

S  
Source term subvector S before eliminating off-diagonal blocks.

**Output**

A, B, C  
Coefficient submatrices A, B, and C after eliminating off-diagonal blocks.

S  
Source term subvector S after eliminating off-diagonal blocks.

**Description**

Subroutine BCELIM eliminates the off-diagonal coefficient submatrices that result from the application of three-point boundary conditions. This is necessary when three-point gradients are specified in the coordinate or normal direction, and when linear extrapolation is used. The procedure is described in Section 7.2.1 of Volume 1.

**Remarks**

1. Subroutines SGEFA and SGESL are Cray LINPACK routines. In general terms, if the Fortran arrays A and B represent A and B, where A is a square N by N matrix and B is a matrix (or vector) with NCOL columns, and if the leading dimension of the Fortran array A is LDA, then the Fortran sequence

```fortran
    call sgefa (a,lda,n,ipvt,info)
    do 10 j = 1,ncol
        call sgesl (a,lda,n,ipvt,b(1,j),0)
    continue
```

computes \( A^{-1}B \), storing the result in B.
Subroutine BCF (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCFLIN, BCMET</td>
<td>Compute user-written boundary conditions.</td>
</tr>
</tbody>
</table>

**Input**

- **DEL**
  - Computational grid spacing in sweep direction.
- **IBASE, ISTEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IBC, FBC**
  - Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to \( N_x \) corresponding to the \( N_x \) conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
- **IBOUND**
  - Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
- **IEQ**
  - Boundary condition equation number.
- **IHSTAG**
  - Flag for constant stagnation enthalpy option.
- **IMIN, IMAX**
  - Minimum and maximum indices in the sweep direction.
- **ISWEEP**
  - Current ADI sweep number.
- **ISWIRL**
  - Flag for swirl in axisymmetric flow.
- **IV**
  - Index in the "vectorized" direction, i.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, \( J^{-1} \).
- **NOUT**
  - Unit number for standard output.
- **NK, NRU, NRV, NRW, NET**
  - Array indices associated with the dependent variables \( \rho, \rho u, \rho v, \rho w, \) and \( E_r \).
- **NVD, NPTSD**
  - Leading two dimensions for the arrays A, B, C, and S.

**Output**

- **A, B, C**
  - Coefficient submatrices A, B, and C at boundary IBOUND (row IEQ only).
- **S**
  - Source term subvector S at boundary IBOUND (element IEQ only).

**Description**

Subroutine BCF computes coefficients and source terms for user-written boundary conditions of the form \( \Delta F = 0, F = f, \partial F/\partial \phi = f, \) and \( \nabla F \cdot \vec{n} = f. \) The values of \( F \) and its derivatives with respect to the dependent variables must be supplied by the user-written subroutine BCFLIN. The linearized equations for these types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections expand these generalized equations in detail.\(^8\)

---

\(^8\) In the following description, for the first ADI sweep the dependent variables should have the superscript \( * \), representing the intermediate solution, and for the second ADI sweep they should have the superscript \( n \), representing
No Change From Initial Conditions, $\Delta F = 0$

Applying equation (6.3) of Volume 1, and noting that $\frac{\partial g}{\partial Q} = J \frac{\partial g}{\partial Q}$, we get simply

$$J_{i,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta E_T \right]_{i,j} = 0$$

**Specified Value, $F = f$**

Applying equation (6.5) of Volume 1,

$$J_{i,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta E_T \right]_{i,j} = f_{i,j}^{n+1} - F_{i,j}^n$$

**Specified Two-Point Gradient in Coordinate Direction, $\partial F/\partial \phi = f$**

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta E_T \right]_{1,j} +$$

$$J_{2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta E_T \right]_{2,j} =$$

$$(\Delta \xi) f_{1,j}^{n+1} + F_{1,j}^n - F_{2,j}^n$$

At the $\xi = 1$ boundary,

$$-J_{N_1-1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta E_T \right]_{N_1-1,j} +$$

$$J_{N_1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta E_T \right]_{N_1,j} =$$

$$(\Delta \xi) f_{N_1,j}^{n+1} + F_{N_1-1,j}^n - F_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

**Specified Three-Point Gradient in Coordinate Direction, $\partial F/\partial \phi = f$**

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,
\[ -3J_{1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{1,j} + \]

\[ 4J_{2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{2,j} - \]

\[ J_{3,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{3,j} = \]

\[ 2(\Delta \xi)^{n+1}_{1,j} + 3F_{1,j}^n - 4F_{2,j}^n + F_{3,j}^n \]

At the \( \xi = 1 \) boundary,

\[ J_{N_1 - 2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{N_1 - 2,j} - \]

\[ 4J_{N_1 - 1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{N_1 - 1,j} + \]

\[ 3J_{N_1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{N_1,j} = \]

\[ 2(\Delta \xi)^{n+1}_{N_1,j} - F_{N_1 - 2,j}^n + 4F_{N_1 - 1,j}^n - 3F_{N_1,j}^n \]

Analogous equations can easily be written for the \( \eta \) boundaries.

*Specified Two-Point Gradient in Normal Direction, \( \nabla F \cdot \hat{n} = f \)

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using two-point one-sided differencing,

\[ -J_{1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{1,j} + \]

\[ J_{2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^n_{2,j} = \]

\[ \frac{\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{1,j}}{m_{1,j}} \delta_n F_{1,j}^n \right] + F_{1,j}^n - F_{2,j}^n \]

where

\[ m = \sqrt{\xi_x^2 + \xi_y^2} \]

and \( \delta_n \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,
Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point Gradient in Normal Direction, \( \nabla F \cdot \hat{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[
-3J_{1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right] +
\]

\[
4J_{2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right] -
\]

\[
J_{3,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right] =
\]

\[
\frac{2\Delta \xi}{m_{1,j}} \left[ f^{n+1}_{1,j} - \frac{(\xi_x n_x + \xi_y n_y)\eta_{1,j}}{m_{1,j}} \delta_{\eta} F_{1,j} \right] + 3F_{1,j}^n - 4F_{2,j}^n + F_{3,j}^n
\]

where

\[ m = \sqrt{\xi_x^2 + \xi_y^2} \]

and \( \delta_{\eta} \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
J_{N_1-2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right] -
\]

\[
4J_{N_1-1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right] +
\]

\[
J_{N_1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right] =
\]

\[
\frac{2\Delta \xi}{m_{N_1,j}} \left[ f^{n+1}_{N_1,j} - \frac{(\xi_x n_x + \xi_y n_y)\eta_{N_1,j}}{m_{N_1,j}} \delta_{\eta} F_{N_1,j} \right] - F_{N_1-2,j}^n + 4F_{N_1-1,j}^n - 3F_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.
Linear Extrapolation

Applying equation (6.14) of Volume 1 at the \( \xi = 0 \) boundary,

\[
J_{1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^{n}_{1,j} -
\]

\[
2J_{2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^{n}_{2,j} +
\]

\[
J_{3,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^{n}_{3,j} =
\]

\[- F_{1,j}^{n} + 2F_{2,j}^{n} - F_{3,j}^{n},\]

At the \( \xi = 1 \) boundary,

\[
J_{N_1-2,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^{n}_{N_1-2,j} -
\]

\[
2J_{N_1-1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^{n}_{N_1-1,j} +
\]

\[
J_{N_1,j} \left[ \frac{\partial F}{\partial \rho} \Delta \hat{\rho} + \frac{\partial F}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial F}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial F}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial F}{\partial E_T} \Delta \hat{E}_T \right]^{n}_{N_1,j} =
\]

\[- F_{N_1-2,j}^{n} + 2F_{N_1-1,j}^{n} - F_{N_1,j}^{n}.\]

Analogous equations can easily be written for the \( \eta \) boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.

2. An error message is generated and execution is stopped if a non-existent user-written boundary condition is specified.
Subroutine BCFLIN (IBC,IEQ,IBOUND,IMIN,IMAX,F,DFDRHO,DFDRU,DFDRV,DFDRW,DFDET,FBCM,FBCP,FBC)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCF</td>
<td></td>
<td>User-supplied routine for linearization of user-supplied boundary conditions.</td>
</tr>
</tbody>
</table>

### Input

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBASE, ISTEP</td>
<td>Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.</td>
</tr>
<tr>
<td>IBC</td>
<td>Mean flow boundary condition types for current sweep direction, specified as IBC(I,J), where I runs from 1 to N_eq, corresponding to the N_eq conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.</td>
</tr>
<tr>
<td>IBOUND</td>
<td>Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.</td>
</tr>
<tr>
<td>IEQ</td>
<td>Boundary condition equation number.</td>
</tr>
<tr>
<td>IMIN, IMAX</td>
<td>Minimum and maximum indices in the sweep direction.</td>
</tr>
<tr>
<td>ISWEEP</td>
<td>Current ADI sweep number.</td>
</tr>
<tr>
<td>I1, I2</td>
<td>Grid indices i and j, in the $\xi$ and $\eta$ directions.</td>
</tr>
<tr>
<td>N1P</td>
<td>Parameter specifying the dimension size in the $\xi$ direction.</td>
</tr>
</tbody>
</table>

### Output

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFDRHO, DFDRU, DFDRV, DFDRW, DFDET</td>
<td>Three-element arrays, specified as DFDRHO(IW), etc., giving the values of $\partial F/\partial \rho$, $\partial F/\partial (\rho u)$, $\partial F/\partial (\rho v)$, $\partial F/\partial (\rho w)$, and $\partial F/\partial E_r$.</td>
</tr>
<tr>
<td>F</td>
<td>A three-element array specified as F(IW) giving the value of the function $F$ at the boundary (IW = 1), at the first point away from the boundary (IW = 2), and at the second point away from the boundary (IW = 3). Values at IW = 3 are not needed for boundary condition types 91, 92, or −92. Values at IW = 2 are not needed for boundary condition type 91.</td>
</tr>
<tr>
<td>FBC</td>
<td>Boundary condition values for current sweep direction, specified as FBC(I,J), where I runs from 1 to N_eq, corresponding to the N_eq conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries. This is only needed if values for GBC1 or GBC2, or FBC1 or FBC2, are not specified in the input namelist BC.</td>
</tr>
<tr>
<td>FBCM, FBCP</td>
<td>Boundary condition values on the boundary, at the grid points &quot;below&quot; and &quot;above&quot; the current boundary point. These are only needed for boundary condition types ±93.</td>
</tr>
</tbody>
</table>

### Description

Subroutine BCFLIN is a user-written routine used in conjunction with subroutine BCF for user-written boundary conditions of the form $\Delta F = 0$, $F = f$, $\partial F/\partial \phi = f$, and $\nabla F \cdot \hat{n} = f$. BCFLIN supplies the values of $F$ and its derivatives with respect to the dependent variables, which are required for writing the linearized form of the boundary condition.
The version of BCFLIN supplied with Proteus makes BCF equivalent to BCTEMP, except for the total temperature options in BCTEMP. Thus \( F = T \), \( \partial F / \partial \rho = \partial T / \partial \rho \), etc., where \( T \) and its derivatives with respect to the dependent variables are computed using the perfect gas equation of state. (See Section 4.3 of Volume 1.) This version of BCFLIN is intended as an example for use in coding boundary conditions not already available.

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. The capability of specifying FBC as an output variable may be useful in writing time-dependent boundary conditions. It also may be used when specifying boundary conditions involving derivatives in both coordinate directions. In this case, the derivatives in the non-sweep direction may be lagged one time step and treated as source terms.
Subroutine BCGEN (A,B,C,S,METX,METY,METT,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BVUP</td>
<td>BCDENS</td>
<td>Manage computation of boundary conditions.</td>
</tr>
<tr>
<td>EXEC</td>
<td>BCDENS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCPRES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCTEMP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCUVEL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCVDIR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCVVEL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCWVEL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISRCHEQ</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- A, B, C
- FBC1, FBC2
  - Point-by-point mean flow boundary condition values for the \(\xi\) and \(\eta\) directions.
- IBC1, IBC2
  - Point-by-point mean flow boundary condition types for the \(\xi\) and \(\eta\) directions.
- IDEBUG
  - Debug flags.
- IPRT1A, IPRT2A
  - Indices for printout in the \(\xi\) and \(\eta\) directions.
- ISWEEP
  - Current ADI sweep number.
- IT
  - Current time step number \(n\).
- IV
  - Index in the "vectorized" direction, \(i\).
- I1, I2
  - Grid indices \(i\) and \(j\), in the \(\xi\) and \(\eta\) directions.
- METX, METY, METT
  - Derivatives of sweep direction computational coordinate with respect to \(x, y\) (or \(r\) if axisymmetric), and \(t\).
- NBC
  - Dimensioning parameter specifying number of boundary conditions per equation.
- NEQ
  - Number of coupled equations being solved, \(N_e\).
- NOUT
  - Unit number for standard output.
- NPRT1, NPRT2
  - Total number of indices for printout in the \(\xi\) and \(\eta\) directions.
- NVD, NPTSD
  - Leading two dimensions for the arrays A, B, C, and S.
- N1, N2
  - Number of grid points \(N_1\) and \(N_2\), in the \(\xi\) and \(\eta\) directions.
- S
  - Source term subvector S.

**Output**

- IBC, FBC
  - Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to \(N_e\), corresponding to the \(N_e\) conditions needed, and \(J = 1\) or 2, corresponding to the lower and upper boundaries.
- IBOUND
  - Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
IEQ
IMIN, IMAX

Boundary condition equation number, from 1 to \(N_e\).
Minimum and maximum indices in the sweep direction.

Description

Subroutine BCGEN manages the computation of coefficients and source terms for the mean flow boundary conditions. It first loads the NEQ boundary condition types and values from the input arrays IBC1 and FBC1, or IBC2 and FBC2, depending on the ADI sweep, into the arrays IBC and FBC. This was done so that the BC routines could be non-sweep dependent. Next the coefficient submatrices and source term subvectors at the two boundaries in the current sweep direction are initialized to zero. And finally, the appropriate BC routine is called, depending on the input boundary condition type, for each of the NEQ boundary conditions at each boundary in the sweep direction.

Remarks

1. An error message is generated and execution is stopped if any of the non-existent boundary condition types 80-89 is specified, or if the boundary condition type is less than 0 or greater than 99.
2. The Cray search routine ISRCHEQ is used in determining the grid locations for debug printout.
3. This subroutine generates the output for the IDEBUG(3) option.
Subroutine BCGRAD (F,I,DFD1,DFD2)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCDENS</td>
<td></td>
<td>Compute gradients with respect to $\xi$ and $\eta$.</td>
</tr>
<tr>
<td>BCF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCPRES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCTEMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCUVEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCVDIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCVVEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCWVEL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- **DXI, DETA**: Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **F**: A two-dimensional array, specified as $F(I,J)$, containing the function $f$ whose gradient is to be computed. The subscripts $I$ and $J$ run from 1 to $N_1$ and $N_2$, respectively.
- **I**: Current grid point index in the current sweep direction.
- **ISWEEP**: Current ADI sweep number.
- **I1, I2**: Grid indices $i$ and $j$ in the $\xi$ and $\eta$ directions.
- **N1, N2**: Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.

**Output**

- **DFD1, DFD2**: First derivatives of $f$ with respect to $\xi$ and $\eta$.

**Description**

Subroutine BCGRAD computes first derivatives of the function $f$, with respect to $\xi$ and $\eta$, at the current grid point in the ADI sweep direction. At interior points, the centered difference formula presented in Section 5.0 of Volume 1 is used. For derivatives with respect to $\xi$,

$$
\left( \frac{\partial f}{\partial \xi} \right)_{i,j} \approx \frac{f_{i+1,j} - f_{i-1,j}}{\Delta \xi}
$$

An analogous formula is used for $\eta$ derivatives.

At boundary points three-point one-sided formulas are used.

$$
\left( \frac{\partial f}{\partial \xi} \right)_{1,j} \approx \frac{1}{2\Delta \xi} (-3f_{1,j} + 4f_{2,j} - f_{3,j})
$$

$$
\left( \frac{\partial f}{\partial \xi} \right)_{N_1,j} \approx \frac{1}{2\Delta \xi} (f_{N_1-2,j} - 4f_{N_1-1,j} + f_{N_1,j})
$$

Again, analogous formulas are used for $\eta$ derivatives.
Subroutine BCMET (I,FM0,FM1,FM2)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCDENS</td>
<td></td>
<td>Compute various metric functions for normal gradient boundary conditions.</td>
</tr>
<tr>
<td>BCF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCPRES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCTEMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCUVEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCVDIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCWVEL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**
- ETAX, ETAY: Metric coefficients $\eta_x$ and $\eta_y$ (or $\eta$, if axisymmetric.)
- I: Current grid point index in the current sweep direction.
- ISWEEP: Current ADI sweep number.
- I1, I2: Grid indices $i$ and $j$ in the $\xi$ and $\eta$ directions.
- XI, XIX, XIY: Metric coefficients $\xi_x$ and $\xi_y$ (or $\xi$, if axisymmetric.)

**Output**
- FM0, FM1, FM2: Various metric functions used for normal derivative boundary conditions.

**Description**
Subroutine BCMET computes metric functions used for normal gradient boundary conditions. For the first ADI sweep,

\[
\begin{align*}
FM0 &= \sqrt{x_{\xi x}^2 + x_{\eta y}^2} \\
FM1 &= 0 \\
FM2 &= \xi_x \eta_x + \xi_y \eta_y
\end{align*}
\]

And for the second sweep,

\[
\begin{align*}
FM0 &= \sqrt{\eta_{\xi x}^2 + \eta_{\eta y}^2} \\
FM1 &= \xi_x \eta_x + \xi_y \eta_y \\
FM2 &= 0
\end{align*}
\]
**Subroutine BCPRES (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NVD,NPTSD)**

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCGRAD</td>
<td>Compute pressure boundary conditions.</td>
</tr>
</tbody>
</table>

**Input**

- **CP, CV**: Specific heats $c_p$ and $c_v$ at time level $n$.
- **DEL**: Computational grid spacing in sweep direction.
- **DPDRHO, DPDRU, DPDRV, DPDRW, DPDET**: Derivatives $\partial p/\partial \rho$, $\partial p/\partial (\rho u)$, $\partial p/\partial (\rho v)$, $\partial p/\partial (\rho w)$, and $\partial p/\partial E_r$.
- **DTDRHO, DTDRU, DTDRV, DTDRW, DTDET**: Derivatives $\partial T/\partial \rho$, $\partial T/\partial (\rho u)$, $\partial T/\partial (\rho v)$, $\partial T/\partial (\rho w)$, and $\partial T/\partial E_r$.
- **GC**: Proportionality factor $g_c$ in Newton's second law.
- **IBASE, ISTEP**: Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IBC, FBC**: Mean flow boundary condition types and values for current sweep direction, specified as $IBC(I,J)$ and $FBC(I,J)$, where $I$ runs from 1 to $N_q$, corresponding to the $N_q$ conditions needed, and $J = 1$ or 2, corresponding to the lower and upper boundaries.
- **IBOUND**: Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
- **IEQ**: Boundary condition equation number.
- **IHSTAG**: Flag for constant stagnation enthalpy option.
- **IMIN, IMAX**: Minimum and maximum indices in the sweep direction.
- **ISWEEP**: Current ADI sweep number.
- **ISWIRL**: Flag for swirl in axisymmetric flow.
- **IV**: Index in the "vectorized" direction, $i$.
- **JI**: Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.
- **NOUT**: Unit number for standard output.
- **NR, NRU, NRV, NRW, NET**: Array indices associated with the dependent variables $\rho$, $\rho u$, $\rho v$, $\rho w$, and $E_r$.
- **NVD, NPTSD**: Leading two dimensions for the arrays $A$, $B$, $C$, and $S$.
- **P, T**: Static pressure $p$ and temperature $T$ at time level $n$.
- **PR**: Reference pressure $p_r$.
- **RGAS**: Gas constant $R$.
- **RHO, U, V, W**: Static density $\rho$, and velocities $u$, $v$, and $w$, at time level $n$.
- **RHOR, UR**: Reference density $\rho_r$ and velocity $u_r$.

**Output**

- **A, B, C**: Coefficient submatrices $A$, $B$, and $C$ at boundary $IBOUND$ (row $IEQ$ only).
Source term subvector \( S \) at boundary \( \text{IBOUND} \) (element IEQ only).

**Description**

Subroutine BCPRES computes coefficients and source terms for pressure boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular pressure boundary conditions in Proteus.\(^9\)

**No Change From Initial Conditions, \( \Delta p = 0 \)**

Applying equation (6.3) of Volume 1, and noting that \( \partial g/\partial \dot{Q} = J \partial g/\partial \dot{Q} \), we get simply

\[
J_{i,j} \left[ \frac{\partial p}{\partial \rho} \Delta \dot{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\dot{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\dot{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\dot{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \dot{E}_T \right]_{i,j}^n = 0
\]

The derivatives \( \partial p/\partial \rho, \partial p/\partial (\rho u), \) etc., depend on the equation of state. They are defined for a perfect gas in Section 4.3 of Volume 1.

**Specified Static Pressure, \( p = f \)**

Applying equation (6.5) of Volume 1,

\[
J_{i,j} \left[ \frac{\partial p}{\partial \rho} \Delta \dot{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\dot{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\dot{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\dot{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \dot{E}_T \right]_{i,j}^n = f_{i,j} + \frac{1}{\rho \mu_T^2} \frac{\partial \rho c}{\partial \rho} - p_{i,j}
\]

**Specified Two-Point Pressure Gradient in Coordinate Direction, \( \partial p/\partial \phi = f \)**

Applying equation (6.8) of Volume 1 at the \( \xi = 0 \) boundary, and using two-point one-sided differencing,

\[
-J_{1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \dot{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\dot{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\dot{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\dot{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \dot{E}_T \right]_{j_1,j}^n +
\]

\[
J_{2,j} \left[ \frac{\partial p}{\partial \rho} \Delta \dot{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\dot{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\dot{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\dot{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \dot{E}_T \right]_{j_2,j}^n = (\Delta \xi v_{1,j}^n + 1 \frac{\partial \rho c}{\partial \rho} + p_{1,j}^n - p_{2,j}^n}
\]

At the \( \xi = 1 \) boundary,

\(^9\) In the following description, for the first ADI sweep the dependent variables should have the superscript \( \ast \), representing the intermediate solution, and for the second ADI sweep they should have the superscript \( \pi \), representing the final solution. For simplicity, however, only the superscript \( \pi \) is used. The superscripts on all other variables are correct as written.
- \[ J_{N_x=1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ J_{N_x,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ ((\Delta \xi f_{N_x,j})^n + 1 \frac{p_{E_c}}{\rho \mu_T^2}) + p_{N_x=1,j}^n - p_{N_x,j}^n \]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point Pressure Gradient in Coordinate Direction, \( \partial p/\partial \phi = f \)**

Applying equation (6.8) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[ -3J_{1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ 4J_{2,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ J_{3,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ 2((\Delta \xi f_{1,j})^n + 1 \frac{p_{E_c}}{\rho \mu_T^2}) + 3p_{1,j}^n - 4p_{2,j}^n + p_{3,j}^n \]

At the \( \xi = 1 \) boundary,

\[ J_{N_x=2,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ 4J_{N_x=1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ 3J_{N_x,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta (\hat{E}_T) \right] \]  

\[ 2((\Delta \xi f_{N_x,j})^n + 1 \frac{p_{E_c}}{\rho \mu_T^2}) - p_{N_x=2,j}^n - 3p_{N_x,j}^n \]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Two-Point Pressure Gradient in Normal Direction, \( \nabla p \cdot \hat{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using two-point one-sided differencing,
\[
-J_{1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right]_{1,j}^n + \\
J_{2,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right]_{2,j}^n = \\
\frac{\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} \frac{p_{8c}}{\rho \mu_i^2} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{1,j}}{m_{1,j}} \delta_n p_{1,j}^n \right] + p_{1,j}^n - p_{2,j}^n,
\]

where

\[
m = \sqrt{\xi_x^2 + \xi_y^2}
\]

and \(\delta_n\) is the centered difference operator presented in Section 5.0 of Volume 1. At the \(\xi = 1\) boundary,

\[
-J_{N_1-1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right]_{N_1-1,j}^n + \\
J_{N_1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right]_{N_1,j}^n = \\
\frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} \frac{p_{8c}}{\rho \mu_i^2} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{N_1,j}}{m_{N_1,j}} \delta_n p_{N_1,j}^n \right] + p_{N_1-1,j}^n - p_{N_1,j}^n,
\]

Analogous equations can easily be written for the \(\eta\) boundaries.

**Specified Three-Point Pressure Gradient in Normal Direction, \(\nabla p \cdot \hat{\eta} = f\)**

Applying equation (6.12a) of Volume 1 at the \(\xi = 0\) boundary, and using three-point one-sided differencing,

\[
-3J_{1,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right]_{1,j}^n + \\
4J_{2,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right]_{2,j}^n = \\
J_{3,j} \left[ \frac{\partial p}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right]_{3,j}^n = \\
\frac{2\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} \frac{p_{8c}}{\rho \mu_i^2} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{1,j}}{m_{1,j}} \delta_n p_{1,j}^n \right] + 3p_{1,j}^n - 4p_{2,j}^n + p_{3,j}^n
\]

where

\[
m = \sqrt{\xi_x^2 + \xi_y^2}
\]

and \(\delta_n\) is the centered difference operator presented in Section 5.0 of Volume 1. At the \(\xi = 1\) boundary,
\[ J_{N_1-2,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ J_{N_1-1,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ J_{N_1,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ \times \frac{2 \Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} \frac{p_{N_1,j}}{\rho_{N_1,j}^2} - \frac{(\xi_{n+1} + \xi_{n+1})_{N_1,j}}{m_{N_1,j}} \delta_{\nu} \right] - p_{N_1-2,j}^n + 4p_{N_1-1,j}^n - 3p_{N_1,j}^n \]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Linear Extrapolation of Static Pressure**

Applying equation (6.14) of Volume 1 at the \( \xi = 0 \) boundary,

\[ J_{1,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ J_{2,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ J_{3,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ - p_{1,j}^n + 2p_{2,j}^n - p_{3,j}^n \]

At the \( \xi = 1 \) boundary,

\[ J_{N_1-2,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ J_{N_1-1,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ J_{N_1,j} \left[ \frac{\partial p}{\partial p} \Delta \hat{\rho} + \frac{\partial p}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p}{\partial E_T} \Delta \hat{E}_T \right] \]

\[ - p_{N_1-2,j}^n + 2p_{N_1-1,j}^n - p_{N_1,j}^n \]

Analogous equations can easily be written for the \( \eta \) boundaries.

**No Change From Initial Conditions for Total Pressure, \( \Delta p_T = 0 \)**

The total pressure is defined as

\[ p_T = p \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{\gamma}{\gamma-1}} \]
Applying equation (6.3) of Volume 1, we get

\[ J_{i,j} \left[ \frac{\partial p_T}{\partial \rho} \Delta \dot{\rho} + \frac{\partial p_T}{\partial (\rho u)} \Delta (\dot{\rho} u) + \frac{\partial p_T}{\partial (\rho v)} \Delta (\dot{\rho} v) + \frac{\partial p_T}{\partial (\rho w)} \Delta (\dot{\rho} w) + \frac{\partial p_T}{\partial E_T} \Delta E_T \right]_{i,j} = 0 \]

where

\[ \frac{\partial p_T}{\partial \rho} = \frac{\partial p}{\partial \rho} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{\gamma}{2}} + p \frac{\gamma}{2} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{1}{2}} \frac{\partial M^2}{\partial \rho} \]

\[ \frac{\partial p_T}{\partial (\rho u)} = \frac{\partial p}{\partial (\rho u)} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{\gamma}{2}} + p \frac{\gamma}{2} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{1}{2}} \frac{\partial M^2}{\partial (\rho u)} \]

\[ \frac{\partial p_T}{\partial (\rho v)} = \frac{\partial p}{\partial (\rho v)} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{\gamma}{2}} + p \frac{\gamma}{2} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{1}{2}} \frac{\partial M^2}{\partial (\rho v)} \]

\[ \frac{\partial p_T}{\partial (\rho w)} = \frac{\partial p}{\partial (\rho w)} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{\gamma}{2}} + p \frac{\gamma}{2} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{1}{2}} \frac{\partial M^2}{\partial (\rho w)} \]

\[ \frac{\partial p_T}{\partial E_T} = \frac{\partial p}{\partial E_T} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{\gamma}{2}} + \frac{\gamma}{2} \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{1}{2}} \frac{\partial M^2}{\partial E_T} \]

The Mach number is defined by

\[ M^2 = \frac{u^2 + v^2 + w^2}{\gamma RT} = \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{\gamma R \rho^2 T} \]

The derivatives \( \partial M^2 / \partial \rho \), etc., can then be derived as

\[ \frac{\partial M^2}{\partial \rho} = -M^2 \left( \frac{2}{\rho} + \frac{1}{T} \frac{\partial T}{\partial \rho} \right) \]

\[ \frac{\partial M^2}{\partial (\rho u)} = \frac{2u}{\rho} - \frac{M^2}{T} \frac{\partial T}{\partial (\rho u)} \]

\[ \frac{\partial M^2}{\partial (\rho v)} = \frac{2v}{\rho} - \frac{M^2}{T} \frac{\partial T}{\partial (\rho v)} \]

\[ \frac{\partial M^2}{\partial (\rho w)} = \frac{2w}{\rho} - \frac{M^2}{T} \frac{\partial T}{\partial (\rho w)} \]

\[ \frac{\partial M^2}{\partial E_T} = -\frac{M^2}{T} \frac{\partial T}{\partial E_T} \]
**Specified Total Pressure, \( p_T = f \)**

Applying equation (6.5) of Volume 1, we get

\[
J_{i,j} \left[ \frac{\partial p_T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial p_T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial p_T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial p_T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial p_T}{\partial E_T} \Delta \hat{E}_T \right]_{i,j}^n =
\]

\[
\frac{f_{i,j}^{n+1} - p_{i,j}^n}{\rho_i u_i^2} - p_{i,j}^n \left[ \left( 1 + \frac{\gamma - 1}{2} M_i^2 \right)^{\frac{\gamma}{\gamma - 1}} \right]_{i,j}^n
\]

where \( p_T, \partial p_T/\partial \rho, \) etc., are defined above as part of the description of the \( \Delta p_T = 0 \) boundary condition.

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent pressure boundary condition is specified.
3. The multiplying factor \( \frac{p g_i}{\rho_i u_i^2} \) that appears with specified values of pressure and pressure gradients is necessary because input values of pressure are nondimensionalized by the reference pressure \( p_r = \rho_r R T_i / g_i \), while internal to the Proteus code itself pressure is nondimensionalized by the normalizing pressure \( p_n = \rho_n u_n^2 \). (See Section 3.1.1 of Volume 2.)
Subroutine BCQ (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCMET</td>
<td>Compute conservation variable boundary conditions.</td>
</tr>
</tbody>
</table>

**Input**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEL</td>
<td>Computational grid spacing $\Delta \xi$ and $\Delta \eta$.</td>
</tr>
<tr>
<td>DXI, DETA</td>
<td>Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.</td>
</tr>
<tr>
<td>IBASE, ISTEP</td>
<td>Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to $N_m$, corresponding to the $N_m$ conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.</td>
</tr>
<tr>
<td>IBC, FBC</td>
<td>Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.</td>
</tr>
<tr>
<td>IBOUND</td>
<td>Boundary condition equation number.</td>
</tr>
<tr>
<td>IEQ</td>
<td>Minimum and maximum indices in the sweep direction.</td>
</tr>
<tr>
<td>IMIN, IMAX</td>
<td>Current ADI sweep number.</td>
</tr>
<tr>
<td>ISWEEP</td>
<td>Flag for swirl in axisymmetric flow.</td>
</tr>
<tr>
<td>* ISWIRL</td>
<td>Index in the &quot;vectorized&quot; direction, $i$,</td>
</tr>
<tr>
<td>IV</td>
<td>Grid indices $i$ and $j$, in the $\xi$ and $\eta$ directions.</td>
</tr>
<tr>
<td>I1, I2</td>
<td>Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.</td>
</tr>
<tr>
<td>JI</td>
<td>Array indices associated with the continuous, $x$-momentum, $y$-momentum (or $r$-momentum if axisymmetric), swirl momentum, and energy equations.</td>
</tr>
<tr>
<td>NC, NXM, NYM, NZM, NEN</td>
<td></td>
</tr>
<tr>
<td>* NOUT</td>
<td>Unit number for standard output.</td>
</tr>
<tr>
<td>NVD, NPTSD</td>
<td>Leading two dimensions for the arrays A, B, C, and S.</td>
</tr>
<tr>
<td>NIP</td>
<td>Parameter specifying the dimension size in the $\xi$ direction.</td>
</tr>
<tr>
<td>RHO, U, V, W, ET</td>
<td>Static density $\rho$, velocities $u$, $v$, and $w$, and total energy $E_T$ at time level $n$.</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C</td>
<td>Coefficient submatrices A, B, and C at boundary IBOUND (row IEQ only).</td>
</tr>
<tr>
<td>S</td>
<td>Source term subvector S at boundary IBOUND (element IEQ only).</td>
</tr>
</tbody>
</table>

**Description**

Subroutine BCQ computes coefficients and source terms for conservation variable boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0.
of Volume 1. The following sections apply these generalized equations to the particular conservation variable boundary conditions in *Proteus*.\(^{10}\)

**No Change From Initial Conditions, \(\Delta Q = 0\)**

Applying equation (6.3) of Volume 1, and noting that \(\partial g/ \partial \hat{Q} = J \partial g/ \partial Q\), we get simply

\[
J_{i,j} \Delta \hat{Q}^\eta_{i,j} = 0
\]

where \(\hat{Q}\) is the element of \(\hat{Q}\) for which this boundary condition is to be applied.

**Specified Conservation Variable, \(Q = f\)**

Applying equation (6.5) of Volume 1,

\[
J_{i,j} \Delta \hat{Q}^\eta_{i,j} = f_{i,j}^{n+1} - Q^\eta_{i,j}
\]

**Specified Two-Point Conservation Variable Gradient in Coordinate Direction, \(\partial Q/ \partial \phi = f\)**

Applying equation (6.8) of Volume 1 at the \(\xi = 0\) boundary, and using two-point one-sided differencing,

\[
-J_{1,j} \Delta \hat{Q}^\eta_{1,j} + J_{2,j} \Delta \hat{Q}^\eta_{2,j} = (\Delta \xi) f^\eta_{1,j} + 1 - Q^\eta_{1,j} - Q^\eta_{2,j}
\]

At the \(\xi = 1\) boundary,

\[
-J_{N_1-1,j} \Delta \hat{Q}^\eta_{N_1-1,j} + J_{N_1,j} \Delta \hat{Q}^\eta_{N_1,j} = (\Delta \xi) f^\eta_{N_1,j} + 1 - Q^\eta_{N_1-1,j} - Q^\eta_{N_1,j}
\]

Analogous equations can easily be written for the \(\eta\) boundaries.

**Specified Three-Point Conservation Variable Gradient in Coordinate Direction, \(\partial Q/ \partial \phi = f\)**

Applying equation (6.8) of Volume 1 at the \(\xi = 0\) boundary, and using three-point one-sided differencing,

\[
-3J_{1,j} \Delta \hat{Q}^\eta_{1,j} + 4J_{2,j} \Delta \hat{Q}^\eta_{2,j} - 3J_{3,j} \Delta \hat{Q}^\eta_{3,j} = 2(\Delta \xi) f^\eta_{1,j} + 1 + 3Q^\eta_{2,j} - 4Q^\eta_{3,j} + Q^\eta_{3,j}
\]

At the \(\xi = 1\) boundary,

\[
J_{N_1-2,j} \Delta \hat{Q}^\eta_{N_1-2,j} - 4J_{N_1-1,j} \Delta \hat{Q}^\eta_{N_1-1,j} + 3J_{N_1,j} \Delta \hat{Q}^\eta_{N_1,j} = 2(\Delta \xi) f^\eta_{N_1,j} + 1 - Q^\eta_{N_1-2,j} + 4Q^\eta_{N_1-1,j} - 3Q^\eta_{N_1,j}
\]

Analogous equations can easily be written for the \(\eta\) boundaries.

**Specified Two-Point Conservation Variable Gradient in Normal Direction, \(\nabla Q \cdot \mathbf{n} = f\)**

Applying equation (6.12a) of Volume 1 at the \(\xi = 0\) boundary, and using two-point one-sided differencing,

\[
10 \text{ In the following description, for the first ADI sweep the dependent variables should have the superscript } *, \text{ representing the intermediate solution, and for the second ADI sweep they should have the superscript } n, \text{ representing the final solution. For simplicity, however, only the superscript } n \text{ is used. The superscripts on all other variables are correct as written.}
\]
where \( \delta_x = \sqrt{x^2 + y^2} \)

and \( \delta_x \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
- J_{N_1 - 1,j} \Delta \hat{Q}_{N_1 - 1,j}^n + J_{N_1,j} \Delta \hat{Q}_{N_1,j}^n - J_{N_1,j} \Delta \hat{Q}_{N_1,j}^n = \frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^n + 1 - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{N_1,j}}{m_{N_1,j}} \delta_x Q^n_{N_1,j} \right] + Q^n_{N_1 - 1,j} - Q^n_{N_1,j}
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point Conservation Variable Gradient in Normal Direction, \( \nabla \Omega \cdot \hat{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[
- 3J_{1,j} \Delta \hat{Q}_{1,j}^n + 4J_{2,j} \Delta \hat{Q}_{2,j}^n - J_{3,j} \Delta \hat{Q}_{3,j}^n = \frac{2\Delta \xi}{m_{1,j}} \left[ f_{1,j}^n + 1 - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{1,j}}{m_{1,j}} \delta_x Q^n_{1,j} \right] + 3Q^n_{1,j} - 4Q^n_{2,j} + Q^n_{3,j}
\]

where \( \delta_x = \sqrt{x^2 + y^2} \)

and \( \delta_x \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
J_{N_1 - 2,j} \Delta \hat{Q}_{N_1 - 2,j}^n - 4J_{N_1 - 1,j} \Delta \hat{Q}_{N_1 - 1,j}^n + 3J_{N_1,j} \Delta \hat{Q}_{N_1,j}^n = \frac{2\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^n + 1 - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{N_1,j}}{m_{N_1,j}} \delta_x Q^n_{N_1,j} \right] - Q^n_{N_1 - 2,j} + 4Q^n_{N_1 - 1,j} - 3Q^n_{N_1,j}
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Linear Extrapolation of Conservation Variable**

Applying equation (6.14) of Volume 1 at the \( \xi = 0 \) boundary,

\[
J_{1,j} \Delta \hat{Q}_{1,j}^n - 2J_{2,j} \Delta \hat{Q}_{2,j}^n + J_{3,j} \Delta \hat{Q}_{3,j}^n = -Q^n_{1,j} + 2Q^n_{2,j} - Q^n_{3,j}
\]

At the \( \xi = 1 \) boundary,

\[
J_{N_1 - 2,j} \Delta \hat{Q}_{N_1 - 2,j}^n - 2J_{N_1 - 1,j} \Delta \hat{Q}_{N_1 - 1,j}^n + J_{N_1,j} \Delta \hat{Q}_{N_1,j}^n = -Q^n_{N_1 - 2,j} + 2Q^n_{N_1 - 1,j} - Q^n_{N_1,j}
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent conservation variable boundary condition is specified.
Subroutine BCSET

Called by Calls Purpose

MAIN Set various boundary condition parameters and flags.

<table>
<thead>
<tr>
<th>Input</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>* GBC1, GBC2</td>
<td></td>
<td>Surface mean flow boundary condition values for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>* GBCT1, GBCT2</td>
<td></td>
<td>Surface ( k-\varepsilon ) boundary condition values for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>* GTBC1, GTBC2</td>
<td></td>
<td>Time-dependent surface mean flow boundary condition values for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>* IHSTAG</td>
<td></td>
<td>Flag for constant stagnation enthalpy option.</td>
</tr>
<tr>
<td>* ISWIRL</td>
<td></td>
<td>Flag for swirl in axisymmetric flow.</td>
</tr>
<tr>
<td>ITDBC</td>
<td></td>
<td>Flag for time-dependent mean flow boundary conditions; 0 if all boundary conditions are steady, 1 if any general unsteady boundary conditions are used, 2 if only steady and time-periodic boundary conditions are used.</td>
</tr>
<tr>
<td>* ITURB</td>
<td></td>
<td>Flag for turbulent flow option.</td>
</tr>
<tr>
<td>* JBC1, JBC2</td>
<td></td>
<td>Surface mean flow boundary condition types for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>* JBCT1, JBCT2</td>
<td></td>
<td>Surface ( k-\varepsilon ) boundary condition types for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>* JTBC1, JTBC2</td>
<td></td>
<td>Flags for type of time dependency for mean flow boundary conditions in the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>* KBC1, KBC2</td>
<td></td>
<td>Boundary types for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>NBC</td>
<td></td>
<td>Dimensioning parameter specifying number of boundary conditions per equation.</td>
</tr>
<tr>
<td>NEQ</td>
<td></td>
<td>Number of coupled equations being solved, ( N_{eq} ).</td>
</tr>
<tr>
<td>* NOUT</td>
<td></td>
<td>Unit number for standard output.</td>
</tr>
<tr>
<td>* NTBC</td>
<td></td>
<td>Number of values in tables for general unsteady boundary conditions.</td>
</tr>
<tr>
<td>* NTBCA</td>
<td></td>
<td>Time levels at which general unsteady boundary conditions are specified.</td>
</tr>
<tr>
<td>* N1, N2</td>
<td></td>
<td>Number of grid points ( N_1 ) and ( N_2 ), in the ( \xi ) and ( \eta ) directions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FBC1, FBC2</td>
<td></td>
<td>Point-by-point mean flow boundary condition values for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>FBCT1, FBCT2</td>
<td></td>
<td>Point-by-point ( k-\varepsilon ) boundary condition values for the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>IBCELM</td>
<td></td>
<td>Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the ( \xi ) and ( \eta ) directions at either boundary; 0 if elimination is not necessary, 1 if it is.</td>
</tr>
</tbody>
</table>
Description

Subroutine BCSET sets various boundary condition parameters and flags. If boundary types are specified with the KBC meta flags, the appropriate surface boundary condition types are loaded into the JBC arrays. Special flags are set if spatially periodic boundary conditions are being used. BCSET also sets NPT1 and NPT2, the number of grid points in each ADI sweep direction to be used in computing coefficients and source terms. For spatially periodic boundary conditions in the $\xi$ direction, $NPT1 = N1 + 1$. Similarly, for spatially periodic boundary conditions in the $\eta$ direction, $NPT2 = N2 + 1$. This is done in order to use central differences at the periodic boundary. (See Section 7.2.2 of Volume 1.)

Next, if the boundary types are being specified using the KBC meta flags, the appropriate JBC mean flow boundary condition parameters are defined. Then, if the mean flow boundary conditions are being specified using the JBC and GBC parameters (or the KBC meta flags), the appropriate point-by-point boundary condition types and values (the IBC and FBC parameters) are loaded with the JBC and GBC values.

If three-point mean flow boundary conditions are being used at a boundary, a flag is set for eliminating the resulting off-diagonal coefficient submatrix. If gradient (two-point or three-point) or extrapolation mean flow boundary conditions are used during the first sweep, a flag is set for updating the $\xi$ boundary values after the second sweep.

Next, for turbulent flow using the $k$-$\varepsilon$ model, if the $k$-$\varepsilon$ boundary conditions are being specified using the JBCT and GBCT parameters, the appropriate point-by-point boundary condition types and values (the IBCT and FBCT parameters) are loaded with the JBCT and GBCT values.

And finally, the input boundary condition parameters are then written to the standard output file.

Remarks

1. An error message is generated and execution is stopped if an invalid boundary type is specified with the KBC meta flags.
Subroutine BCTEMP (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCGRAD</td>
<td>Compute temperature boundary conditions.</td>
</tr>
<tr>
<td></td>
<td>BCMET</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- CP, CV: Specific heats $c_p$ and $c_v$ at time level $n$.
- DEL: Computational grid spacing in sweep direction.
- DTDRHO, DTDRU, DTDVR, DTDRW, DTDET: Derivatives $\partial T/\partial \rho$, $\partial T/\partial (\rho u)$, $\partial T/\partial (\rho v)$, $\partial T/\partial (\rho w)$, and $\partial T/\partial E_T$.
- IBASE, ISTEP: Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- IBC, FBC: Mean flow boundary condition types and values for current sweep direction, specified as $IBC(I,J)$ and $FBC(I,J)$, where $I$ runs from 1 to $N_e$, corresponding to the $N_e$ conditions needed, and $J = 1$ or 2, corresponding to the lower and upper boundaries.
- IBOUND: Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
- IEQ: Boundary condition equation number.
- *IHSTAG: Flag for constant stagnation enthalpy option.
- IMIN, IMAX: Minimum and maximum indices in the sweep direction.
- ISWEEP: Current ADI sweep number.
- *ISWIRL: Flag for swirl in axisymmetric flow.
- IV: Index in the "vectorized" direction, $i$.
- JI: Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.
- *NOUT: Unit number for standard output.
- NR, NRU, NRV, NRW, NET: Array indices associated with the dependent variables $\rho$, $\rho u$, $\rho v$, $\rho w$, and $E_T$.
- NVD, NPTSD: Leading two dimensions for the arrays A, B, C, and S.
- P, T: Static pressure $p$ and temperature $T$ at time level $n$.
- RGAS: Gas constant $R$.
- RHO, U, V, W: Static density $\rho$, and velocities $u$, $v$, and $w$, at time level $n$.

**Output**

- A, B, C: Coefficient submatrices A, B, and C at boundary IBOUND (row IEQ only).
- S: Source term subvector S at boundary IBOUND (element IEQ only).

**Description**

Subroutine BCTEMP computes coefficients and source terms for temperature boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0.
of Volume 1. The following sections apply these generalized equations to the particular temperature boundary conditions in *Proteus*.\(^{11}\)

**No Change From Initial Conditions, \(\Delta T = 0\)**

Applying equation (6.3) of Volume 1, and noting that \(\partial g / \partial Q = J \partial g / \partial Q\), we get simply

\[
J_{i,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{i,j}^n = 0
\]

The derivatives \(\partial T / \partial \rho, \partial T / \partial (\rho u), \) etc., depend on the equation of state. They are defined for a perfect gas in Section 4.3 of Volume 1.

**Specified Static Temperature, \(T = f\)**

Applying equation (6.5) of Volume 1,

\[
J_{i,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{i,j}^n = f_{i,j}^{n+1} - T_{i,j}^n
\]

**Specified Two-Point Temperature Gradient in Coordinate Direction, \(\partial T / \partial \eta = f\)**

Applying equation (6.8) of Volume 1 at the \(\xi = 0\) boundary, and using two-point one-sided differencing,

\[
-J_{1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{1,j}^n +
\]

\[
J_{2,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{2,j}^n =
\]

\[
(\Delta \xi)^{n+1}_{1,j} + T_{1,j}^n - T_{2,j}^n
\]

At the \(\xi = 1\) boundary,

\[
-J_{N_1 - 1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1 - 1,j}^n +
\]

\[
J_{N_1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1,j}^n =
\]

\[
(\Delta \xi)^{n+1}_{N_1,j} + T_{N_1 - 1,j}^n - T_{N_1,j}^n
\]

Analogous equations can easily be written for the \(\eta\) boundaries.

**Specified Three-Point Temperature Gradient in Coordinate Direction, \(\partial T / \partial \phi = f\)**

Applying equation (6.8) of Volume 1 at the \(\xi = 0\) boundary, and using three-point one-sided differencing,

\[
\text{In the following description, for the first ADI sweep the dependent variables should have the superscript *, representing the intermediate solution, and for the second ADI sweep they should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.}
\]

88 4.0 Proteus Subprograms: BCTEMP Proteus 2-D Programmer’s Reference
At the $\xi = 1$ boundary,

$$-3J_{1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \rho \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{1,j}^n +$$

$$4J_{2,j} \left[ \frac{\partial T}{\partial \rho} \Delta \rho \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{2,j}^n -$$

$$J_{3,j} \left[ \frac{\partial T}{\partial \rho} \Delta \rho \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{3,j}^n =$$

$$2(\Delta \xi)^n J_{1,j}^{n+1} + 3T_{1,j}^n - 4T_{2,j}^n + T_{3,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

Specified Two-Point Temperature Gradient in Normal Direction, $\nabla T \cdot \hat{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \rho \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{1,j}^n +$$

$$J_{2,j} \left[ \frac{\partial T}{\partial \rho} \Delta \rho \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{2,j}^n =$$

$$\frac{\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \frac{(\xi_x n_x + \xi_y n_y)_{1,j}}{m_{1,j}} T_{1,j}^n \right] + T_{1,j}^n - T_{2,j}^n$$

where

$$m = \sqrt{\frac{\Delta x^2}{\Delta y^2}}$$

and $\delta_{\eta}$ is the centered difference operator presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,
\[-J_{N_1-1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1-1,j}^{n} + \]

\[J_{N_1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1,j}^{n} = \]

\[\frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \frac{1}{m_{N_1,j}} \delta_\eta T^\eta_{N_1,j} \right] + T_{N_1-1,j}^{n} - T_{N_1,j}^{n} \]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point Temperature Gradient in Normal Direction, \( \nabla T \cdot \hat{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[-3J_{1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{1,j}^{n} + \]

\[4J_{2,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{2,j}^{n} - \]

\[J_{3,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{3,j}^{n} = \]

\[\frac{2\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \frac{1}{m_{1,j}} \delta_\eta T^\eta_{1,j} \right] + 3T_{1,j}^{n} - 4T_{2,j}^{n} + T_{3,j}^{n} \]

where

\[m = \sqrt{\xi_x^2 + \xi_y^2} \]

and \( \delta_\eta \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[J_{N_1-2,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1-2,j}^{n} = \]

\[4J_{N_1-1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1-1,j}^{n} + \]

\[3J_{N_1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1,j}^{n} = \]

\[\frac{2\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \frac{1}{m_{N_1,j}} \delta_\eta T^\eta_{N_1,j} \right] - T_{N_1-2,j}^{n} + 4T_{N_1-1,j}^{n} - 3T_{N_1,j}^{n} \]

Analogous equations can easily be written for the \( \eta \) boundaries.
**Linear Extrapolation of Static Temperature**

Applying equation (6.14) of Volume 1 at the \( \xi = 0 \) boundary,

\[
J_{1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial^2 T}{\partial E_T} \Delta \hat{E}_T \right]_{1,j}^n =
\]

\[
2J_{2,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial^2 T}{\partial E_T} \Delta \hat{E}_T \right]_{2,j}^n +
\]

\[
J_{3,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial^2 T}{\partial E_T} \Delta \hat{E}_T \right]_{3,j}^n =
\]

\[- T_{1,j}^n + 2T_{2,j}^n - T_{3,j}^n \]

At the \( \xi = 1 \) boundary,

\[
J_{N_1 - 2,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial^2 T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1 - 2,j}^n =
\]

\[
2J_{N_1 - 1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial^2 T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1 - 1,j}^n +
\]

\[
J_{N_1,j} \left[ \frac{\partial T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial^2 T}{\partial E_T} \Delta \hat{E}_T \right]_{N_1,j}^n =
\]

\[- T_{N_1 - 2,j}^n + 2T_{N_1 - 1,j}^n - T_{N_1,j}^n \]

Analogous equations can easily be written for the \( \eta \) boundaries.

**No Change From Initial Conditions for Total Temperature, \( \Delta T_T = 0 \)**

The total temperature is defined as

\[ T_T = T \left( 1 + \frac{\gamma - 1}{2} M^2 \right) \]

Applying equation (6.3) of Volume 1, we get

\[
J_{i,j} \left[ \frac{\partial T_T}{\partial \rho} \Delta \hat{\rho} + \frac{\partial T_T}{\partial (\rho u)} \Delta (\hat{\rho} u) + \frac{\partial T_T}{\partial (\rho v)} \Delta (\hat{\rho} v) + \frac{\partial T_T}{\partial (\rho w)} \Delta (\hat{\rho} w) + \frac{\partial^2 T_T}{\partial E_T} \Delta \hat{E}_T \right]_{i,j}^n = 0
\]

where
The Mach number is defined by

\[ M^2 = \frac{u^2 + v^2 + w^2}{\gamma RT} = \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{\gamma \rho^2 T} \]

The derivatives \( \partial M^2/\partial \rho \), etc., can then be derived as

\[ \frac{\partial M^2}{\partial \rho} = -M^2 \left( \frac{2}{\rho} + \frac{1}{T} \frac{\partial T}{\partial \rho} \right) \]

\[ \frac{\partial M^2}{\partial (\rho u)} = \frac{2u}{\gamma \rho} - \frac{M^2}{T} \frac{\partial T}{\partial (\rho u)} \]

\[ \frac{\partial M^2}{\partial (\rho v)} = \frac{2v}{\gamma \rho} - \frac{M^2}{T} \frac{\partial T}{\partial (\rho v)} \]

\[ \frac{\partial M^2}{\partial (\rho w)} = \frac{2w}{\gamma \rho} - \frac{M^2}{T} \frac{\partial T}{\partial (\rho w)} \]

\[ \frac{\partial M^2}{\partial E_T} = -\frac{M^2}{T} \frac{\partial T}{\partial E_T} \]

Specified Total Temperature, \( T_T = f \)

Applying equation (6.5) of Volume 1, we get

\[ J_{i,j} \left[ \frac{\partial T_T}{\partial \rho} \Delta \rho + \frac{\partial T_T}{\partial (\rho u)} \Delta (\rho u) + \frac{\partial T_T}{\partial (\rho v)} \Delta (\rho v) + \frac{\partial T_T}{\partial (\rho w)} \Delta (\rho w) + \frac{\partial T_T}{\partial E_T} \Delta E_T \right]_{i,j}^n = f_{i,j}^{n+1} - T_{i,j}^n \left( 1 + \frac{\gamma - 1}{2} M^2 \right)_{i,j}^n \]

where \( T_T, \partial T_T/\partial \rho \), etc., are defined above as part of the description of the \( \Delta T_T = 0 \) boundary condition.

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent temperature boundary condition is specified.
Subroutine BCUVEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NVD,NPTSD)

Called by Calls Purpose
BCGEN BCGRAD BCMET Compute x-velocity boundary conditions.

Input

DEL Computational grid spacing in sweep direction.
IBASE, ISTEP Base index and multiplication factor used in computing one-
dimensional index for two-dimensional array.
IBC, FBC Mean flow boundary condition types and values for current sweep
direction, specified as IBC(I,J) and FBC(I,J), where I runs from
1 to Nq, corresponding to the Nq conditions needed, and J = 1 or 2,
corresponding to the lower and upper boundaries.
IBOUND Flag specifying boundary; 1 for lower boundary, 2 for upper
boundary.
IEQ Boundary condition equation number.
IMIN, IMAX Minimum and maximum indices in the sweep direction.
ISWEEP Current ADI sweep number.
IV Index in the "vectorized" direction, i.
JI Inverse Jacobian of the nonorthogonal grid transformation, J^{-1}.
* NOUT Unit number for standard output.
NR, NRU Array indices associated with the dependent variables ρ and ρu.
NVD, NPTSD Leading two dimensions for the arrays A, P, C, and S.
RHO, U Static density ρ and velocity u at time level n.

Output

A, B, C Coefficient submatrices A, B, and C at boundary IBOUND (row
IEQ only).
S Source term subvector S at boundary IBOUND (element IEQ
only).

Description

Subroutine BCUVEL computes coefficients and source terms for x-velocity boundary conditions. The
linearized equations for the various general types of boundary conditions are developed in Section 6.0 of
Volume 1. The following sections apply these generalized equations to the particular x-velocity boundary
conditions in Proteus.12

12 In the following description, for the first ADI sweep the dependent variables should have the superscript *,
representing the intermediate solution, and for the second ADI sweep they should have the superscript n,
representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables
are correct as written.
No Change From Initial Conditions, $\Delta u = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial Q = J \partial g/\partial Q$, we get simply

$$ J_{i,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{i,j} = 0 $$

**Specified x-Velocity, $u = f$**

Applying equation (6.5) of Volume 1,

$$ J_{i,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{i,j} = f^n_{i,j} + 1 - u^n_{i,j} $$

**Specified Two-Point x-Velocity Gradient in Coordinate Direction, $\partial u/\partial \phi = f$**

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$ -J_{1,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{1,j} + J_{2,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{2,j} = (\Delta \hat{\xi}) f^n_{1,j} + 1 + u^n_{1,j} - u^n_{2,j} $$

At the $\xi = 1$ boundary,

$$ -J_{N_1-1,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{N_1-1,j} + J_{N_1,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{N_1,j} = (\Delta \hat{\xi}) f^n_{N_1,j} + 1 + u^n_{N_1-1,j} - u^n_{N_1,j} $$

Analogous equations can easily be written for the $\eta$ boundaries.

**Specified Three-Point x-Velocity Gradient in Coordinate Direction, $\partial u/\partial \phi = f$**

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$ -3J_{1,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{1,j} + 4J_{2,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{2,j} - $$

$$ J_{3,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{3,j} = 2(\Delta \hat{\xi}) f^n_{1,j} + 1 + 3u^n_{1,j} - 4u^n_{2,j} + u^n_{3,j} $$

At the $\xi = 1$ boundary,

$$ J_{N_1-2,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{N_1-2,j} - 4J_{N_1-1,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{N_1-1,j} + $$

$$ 3J_{N_1,j} \left[ -\frac{\mu}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho u) \right]^n_{N_1,j} = 2(\Delta \hat{\xi}) f^n_{N_1,j} + 1 - u^n_{N_1-2,j} + 4u^n_{N_1-1,j} - 3u^n_{N_1,j} $$

Analogous equations can easily be written for the $\eta$ boundaries.
Specified Two-Point x-Velocity Gradient in Normal Direction, $\nabla u \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$- J_{1,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{1,j}^n + J_{2,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{2,j}^n =$$

$$\frac{\Delta \xi}{m_{1,j}} \left[ j_{1,j}^{n+1} - \frac{(\xi x \eta_x + \xi y \eta_y)_{1,j}}{m_{1,j}} \delta_n u_{1,j}^n \right] + u_{1,j}^n - u_{2,j}^n$$

where

$$m = \sqrt{\xi_x^2 + \xi_y^2}$$

and $\delta_n$ is the centered difference operator presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$- J_{N_2 - 1,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{N_2 - 1,j}^n + J_{N_2,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{N_2,j}^n =$$

$$\frac{\Delta \xi}{m_{N_2,j}} \left[ j_{N_2,j}^{n+1} - \frac{(\xi x \eta_x + \xi y \eta_y)_{N_2,j}}{m_{N_2,j}} \delta_n u_{N_2,j}^n \right] + u_{N_2 - 1,j}^n - u_{N_2,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

Specified Three-Point x-Velocity Gradient in Normal Direction, $\nabla u \cdot \vec{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$- 3J_{1,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{1,j}^n + 4J_{2,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{2,j}^n -$$

$$J_{3,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{3,j}^n = \frac{2\Delta \xi}{m_{1,j}} \left[ j_{1,j}^{n+1} - \frac{(\xi x \eta_x + \xi y \eta_y)_{1,j}}{m_{1,j}} \delta_n u_{1,j}^n \right] +$$

$$3u_{1,j}^n - 4u_{2,j}^n + u_{3,j}^n$$

where

$$m = \sqrt{\xi_x^2 + \xi_y^2}$$

and $\delta_n$ is the centered difference operator presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$J_{N_2 - 2,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{N_2 - 2,j}^n - 4J_{N_2 - 1,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{N_2 - 1,j}^n +$$

$$J_{N_2,j} \left[ - \frac{u}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho}u) \right]_{N_2,j}^n = \frac{2\Delta \xi}{m_{N_2,j}} \left[ j_{N_2,j}^{n+1} - \frac{(\xi x \eta_x + \xi y \eta_y)_{N_2,j}}{m_{N_2,j}} \delta_n u_{N_2,j}^n \right] -$$

$$u_{N_2 - 2,j}^n + 4u_{N_2 - 1,j}^n - 3u_{N_2,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.
Linear Extrapolation of \( x\)-Velocity

Applying equation (6.14) of Volume I at the \( \xi = 0 \) boundary,

\[
J_{1,j} \left[ -\frac{u}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho \hat{u}) \right]_{1,j}^n - 2J_{2,j} \left[ -\frac{u}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho \hat{u}) \right]_{2,j}^n + \\
J_{3,j} \left[ -\frac{u}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho \hat{u}) \right]_{3,j}^n = -u_{1,j}^n + 2u_{2,j}^n - u_{3,j}^n
\]

At the \( \xi = 1 \) boundary,

\[
J_{N_1 - 2,j} \left[ -\frac{u}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho \hat{u}) \right]_{N_1 - 2,j}^n - 2J_{N_1 - 1,j} \left[ -\frac{u}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho \hat{u}) \right]_{N_1 - 1,j}^n + \\
J_{N_1,j} \left[ -\frac{u}{\rho} \Delta \hat{p} + \frac{1}{\rho} \Delta (\rho \hat{u}) \right]_{N_1,j}^n = -u_{N_1 - 2,j}^n + 2u_{N_1 - 1,j}^n - u_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.

2. An error message is generated and execution is stopped if a non-existent \( x\)-velocity boundary condition is specified.
Subroutine BCVDIR (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,METX,METY,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCGRAD</td>
<td>Compute normal and tangential velocity boundary conditions.</td>
</tr>
<tr>
<td>BCMET</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input

- **DEL**
  - Computational grid spacing in sweep direction.

- **DXI, DETA**
  - Computational grid spacing $\Delta \xi$ and $\Delta \eta$.

- **IBASE, ISTEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.

- **IBC, FBC**
  - Mean flow boundary condition types and values for current sweep direction, specified as $IBC(I,J)$ and $FBC(I,J)$, where $I$ runs from 1 to $N_{\eta}$, corresponding to the $N_{\eta}$ conditions needed, and $J = 1$ or 2, corresponding to the lower and upper boundaries.

- **IBOUND**
  - Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.

- **IEQ**
  - Boundary condition equation number.

- **IMIN, IMAX**
  - Minimum and maximum indices in the sweep direction.

- **ISWEEP**
  - Current ADI sweep number.

- **ISWIRL**
  - Flag for swirl in axisymmetric flow.

- **IV**
  - Index in the "vectorized" direction, $i$.

- **I1, I2**
  - Grid indices $i$ and $j$, in the $\xi$ and $\eta$ directions.

- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.

- **METX, METY**
  - Derivatives of sweep direction computational coordinate with respect to $x$ and $y$ (or $r$ if axisymmetric.)

- **NOUT**
  - Unit number for standard output.

- **NR, NRU, NRV, NRW**
  - Array indices associated with the dependent variables $\rho$, $\rho u$, $\rho v$, and $\rho w$.

- **NVD, NPTSD**
  - Leading two dimensions for the arrays $A$, $B$, $C$, $S$, $METX$, and $METY$.

- **NIP**
  - Parameter specifying the dimension size in the $\xi$ direction.

- **RHO, U, V, W**
  - Static density $\rho$, and velocities $u$, $v$, and $w$, at time level $n$.

Output

- **A, B, C**
  - Coefficient submatrices $A$, $B$, and $C$ at boundary $IBOUND$ (row IEQ only).

- **S**
  - Source term subvector $S$ at boundary $IBOUND$ (element IEQ only).

Description

Subroutine BCVDIR computes coefficients and source terms for normal and tangential velocity boundary conditions. The linearized equations for the various general types of boundary conditions are
developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular normal and tangential velocity boundary conditions in Proteus.

**Specified Normal Velocity, \( V_n = f \)**

The normal velocity is defined as

\[ V_n = \vec{V} \cdot \vec{n} \]

where \( \vec{n} \) is the unit vector normal to the boundary. For a \( \xi \) boundary,

\[ \vec{n} = \frac{\nabla \xi}{|\nabla \xi|} = \frac{1}{m} \xi_x \vec{i} + \frac{1}{m} \xi_y \vec{j} \]

where

\[ m = \sqrt{\xi_x^2 + \xi_y^2} \]

Therefore, for a \( \xi \) boundary,

\[ V_n = \frac{1}{m} (\xi_x u + \xi_y v) = f \]

Note that the unit vector \( \vec{n} \) is in the direction of increasing \( \xi \). Therefore \( V_n \) is positive in the direction of increasing \( \xi \). Thus, a positive \( V_n \) at \( \xi = 0 \) implies flow into the computational domain, and a positive \( V_n \) at \( \xi = 1 \) implies flow out of the computational domain.

Similarly, for an \( \eta \) boundary,

\[ V_n = \frac{1}{m} (\eta_x u + \eta_y v) = f \]

where

\[ m = \sqrt{\eta_x^2 + \eta_y^2} \]

and \( V_n \) is positive in the direction of increasing \( \eta \).

Applying equation (6.5) of Volume 1, the linearized boundary condition at a \( \xi \) boundary becomes

\[ \frac{J_{i,j}}{m_{i,j}} \left[ \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{p} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{i,j} = f_{i,j}^{n+1} - (V_n)_{i,j} \]

An analogous equation can easily be written for the \( \eta \) boundaries.

---

13 In the following description, for the first ADI sweep the dependent variables should have the superscript *, representing the intermediate solution, and for the second ADI sweep they should have the superscript \( n \), representing the final solution. For simplicity, however, only the superscript \( n \) is used. The superscripts on all other variables are correct as written.
Specified Two-Point Normal Velocity Gradient in Coordinate Direction, $\partial V / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$\frac{J_{1,j}}{m_{1,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{1,j}^n +$$

$$\frac{J_{2,j}}{m_{2,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{2,j}^n =$$

$$(\Delta \xi)_{f_{1,j}}^{n+1} + (V_n)^n_{1,j} - (V_n)^n_{2,j}$$

At the $\xi = 1$ boundary,

$$\frac{J_{N_1,j}}{m_{N_1,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1,j}^n +$$

$$\frac{J_{N_1,j}}{m_{N_1,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1,j}^n =$$

$$(\Delta \xi)_{f_{N_1,j}}^{n+1} + (V_n)^n_{N_1,j} - (V_n)^n_{N_1,j}$$

Analogous equations can easily be written for the $\eta$ boundaries.

Specified Three-Point Normal Velocity Gradient in Coordinate Direction, $\partial V / \partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$- \frac{3}{m_{1,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{1,j}^n +$$

$$4 \frac{J_{2,j}}{m_{2,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{2,j}^n -$$

$$3 \frac{J_{3,j}}{m_{3,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{3,j}^n =$$

$$2(\Delta \xi)_{f_{1,j}}^{n+1} + 3(V_n)^n_{1,j} - 4(V_n)^n_{2,j} + (V_n)^n_{3,j}$$

At the $\xi = 1$ boundary,

$$\frac{J_{N_1-2,j}}{m_{N_1-2,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1-2,j}^n -$$

$$4 \frac{J_{N_1-1,j}}{m_{N_1-1,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1-1,j}^n +$$

$$3 \frac{J_{N_1,j}}{m_{N_1,j}} \left[ - \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1,j}^n =$$

$$2(\Delta \xi)_{f_{N_1,j}}^{n+1} - (V_n)^n_{N_1-2,j} + 4(V_n)^n_{N_1-1,j} - 3(V_n)^n_{N_1,j}$$
Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Two-Point Normal Velocity Gradient in Normal Direction, \( \nabla V_n \cdot \vec{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using two-point one-sided differencing,

\[
- \frac{J_{1,j}}{m_{1,j}} \left[ \frac{- \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\hat{\rho}u) + \frac{\xi_y}{\rho} \Delta (\hat{\rho}v)}{1,j} \right]^n +
\]

\[
\frac{J_{2,j}}{m_{2,j}} \left[ \frac{- \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\hat{\rho}u) + \frac{\xi_y}{\rho} \Delta (\hat{\rho}v)}{2,j} \right]^n =
\]

\[
\frac{\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \frac{\xi_x \eta_x + \xi_y \eta_y}{m_{1,j}} \delta_n(V_n)_{1,j} \right] + (V_n)_{1,j} - (V_n)_{2,j}
\]

where \( \delta_n \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
- \frac{J_{N_1 - 1,j}}{m_{N_1 - 1,j}} \left[ \frac{- \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\hat{\rho}u) + \frac{\xi_y}{\rho} \Delta (\hat{\rho}v)}{N_1 - 1,j} \right]^n +
\]

\[
\frac{J_{N_1,j}}{m_{N_1,j}} \left[ \frac{- \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\hat{\rho}u) + \frac{\xi_y}{\rho} \Delta (\hat{\rho}v)}{N_1,j} \right]^n =
\]

\[
\frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \frac{\xi_x \eta_x + \xi_y \eta_y}{m_{N_1,j}} \delta_n(V_n)_{N_1,j} \right] + (V_n)_{N_1 - 1,j} - (V_n)_{N_1,j}
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point Normal Velocity Gradient in Normal Direction, \( \nabla V_n \cdot \vec{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[
- 3 \frac{J_{1,j}}{m_{1,j}} \left[ \frac{- \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\hat{\rho}u) + \frac{\xi_y}{\rho} \Delta (\hat{\rho}v)}{1,j} \right]^n +
\]

\[
4 \frac{J_{2,j}}{m_{2,j}} \left[ \frac{- \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\hat{\rho}u) + \frac{\xi_y}{\rho} \Delta (\hat{\rho}v)}{2,j} \right]^n -
\]

\[
\frac{J_{3,j}}{m_{3,j}} \left[ \frac{- \frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{\rho} + \frac{\xi_x}{\rho} \Delta (\hat{\rho}u) + \frac{\xi_y}{\rho} \Delta (\hat{\rho}v)}{3,j} \right]^n =
\]

\[
\frac{2\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \frac{\xi_x \eta_x + \xi_y \eta_y}{m_{1,j}} \delta_n(V_n)_{1,j} \right] + (V_n)_{1,j} - 4(V_n)_{2,j} + (V_n)_{3,j}
\]

where \( \delta_n \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,
Analogous equations can easily be written for the \( \eta \) boundaries.

**Linear Extrapolation of Normal Velocity**

Applying equation (6.14) of Volume 1 at the \( \xi = 0 \) boundary,

\[
\frac{J_{N_1-2,j}}{m_{N_1-2,j}} \left[ -\frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{p} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1-2,j}^n
\]

\[
\frac{J_{N_1-1,j}}{m_{N_1-1,j}} \left[ -\frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{p} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1-1,j}^n
\]

\[
\frac{J_{N_1,j}}{m_{N_1,j}} \left[ -\frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{p} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1,j}^n
\]

\[
\frac{2\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \frac{(\xi_x n_x + \xi_y n_y)_{N_1,j}}{m_{N_1,j}} \delta_n (V_n)_{N_1,j}^n \right] - (V_n)_{N_1-2,j}^n + 4(V_n)_{N_1-1,j}^n - 3(V_n)_{N_1,j}^n
\]

At the \( \xi = 1 \) boundary,

\[
\frac{J_{N_1-2,j}}{m_{N_1-2,j}} \left[ -\frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{p} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1-2,j}^n
\]

\[
\frac{J_{N_1-1,j}}{m_{N_1-1,j}} \left[ -\frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{p} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1-1,j}^n
\]

\[
\frac{J_{N_1,j}}{m_{N_1,j}} \left[ -\frac{\xi_x u + \xi_y v}{\rho} \Delta \hat{p} + \frac{\xi_x}{\rho} \Delta (\rho u) + \frac{\xi_y}{\rho} \Delta (\rho v) \right]_{N_1,j}^n
\]

\[
-(V_n)_{N_1-2,j}^n + 2(V_n)_{N_1-1,j}^n - (V_n)_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Tangential Velocity, \( V_\eta = f \)**

For a \( \xi \) boundary, the tangential velocity is the velocity in the \( \eta \) direction, and is given by

\[
V_\eta = V_\eta = \sqrt{u^2 + v^2 - V_n^2}
\]

\[
= \frac{1}{m} (-\xi_x u + \xi_y v)
\]
where

$$m = \sqrt{\xi_x^2 + \xi_y^2}$$

Note that this definition for tangential velocity does not include the swirl velocity \( w \) in axisymmetric flow. Separate boundary conditions should be set for \( w \).

Similarly, for an \( \eta \) boundary,

$$V_t = V_\xi = \frac{1}{m} (\eta_t \mu - \eta_x v)$$

where

$$m = \sqrt{\eta_x^2 + \eta_y^2}$$

Applying equation (6.5) of Volume 1, the linearized boundary condition at a \( \zeta \) boundary becomes

$$\frac{J_{1,j}}{m_{1,j}} \left[ \frac{\xi_x u - \xi_x v}{\rho} \Delta \hat{p} - \frac{\xi_y}{\rho} \Delta (\rho \hat{u}) + \frac{\xi_x}{\rho} \Delta (\rho \hat{v}) \right]_1^{n+1} = f_{1,j} - (V_{\phi})_{1,j}^n$$

An analogous equation can easily be written for the \( \eta \) boundaries.

**Specified Two-Point Tangential Velocity Gradient in Coordinate Direction, \( \partial V_\xi / \partial \phi = f \)**

Applying equation (6.8) of Volume 1 at the \( \zeta = 0 \) boundary, and using two-point one-sided differencing,

$$- \frac{J_{1,j}}{m_{1,j}} \left[ \frac{\xi_x u - \xi_x v}{\rho} \Delta \hat{p} - \frac{\xi_y}{\rho} \Delta (\rho \hat{u}) + \frac{\xi_x}{\rho} \Delta (\rho \hat{v}) \right]_{1,j}^{n+1} + \frac{J_{2,j}}{m_{2,j}} \left[ \frac{\xi_x u - \xi_x v}{\rho} \Delta \hat{p} - \frac{\xi_y}{\rho} \Delta (\rho \hat{u}) + \frac{\xi_x}{\rho} \Delta (\rho \hat{v}) \right]_{2,j}^{n} = (\Delta \zeta)^{n+1} + (V_{\phi})_{1,j}^n - (V_{\phi})_{2,j}^n$$

At the \( \zeta = 1 \) boundary,

$$- \frac{J_{N_1 - 1,j}}{m_{N_1 - 1,j}} \left[ \frac{\xi_x u - \xi_x v}{\rho} \Delta \hat{p} - \frac{\xi_y}{\rho} \Delta (\rho \hat{u}) + \frac{\xi_x}{\rho} \Delta (\rho \hat{v}) \right]_{N_1 - 1,j}^{n} + \frac{J_{N_1,j}}{m_{N_1,j}} \left[ \frac{\xi_x u - \xi_x v}{\rho} \Delta \hat{p} - \frac{\xi_y}{\rho} \Delta (\rho \hat{u}) + \frac{\xi_x}{\rho} \Delta (\rho \hat{v}) \right]_{N_1,j}^{n} = (\Delta \zeta)^{n+1} + (V_{\phi})_{N_1 - 1,j}^n - (V_{\phi})_{N_1,j}^n$$

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point Tangential Velocity Gradient in Coordinate Direction, \( \partial V_\xi / \partial \phi = f \)**

Applying equation (6.8) of Volume 1 at the \( \zeta = 0 \) boundary, and using three-point one-sided differencing,
\begin{align*}
-3 \frac{J_{1,j}}{m_{1,j}} & \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{1,j}^n + \\
4 \frac{J_{2,j}}{m_{2,j}} & \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{2,j}^n - \\
3 \frac{J_{3,j}}{m_{3,j}} & \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{3,j}^n = \\
2(\Delta \xi)_{N_{1,j}}^{n+1} + 3(V_{1\eta})_{N_{1,j}}^n - 4(V_{2\eta})_{N_{1,j}}^n + (V_{3\eta})_{N_{1,j}}^n
\end{align*}

At the \( \xi = 1 \) boundary,
\begin{align*}
& \frac{J_{N_{1} - 2,j}}{m_{N_{1} - 2,j}} \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1} - 2,j}^n - \\
& \frac{J_{N_{1} - 1,j}}{m_{N_{1} - 1,j}} \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1} - 1,j}^n + \\
& \frac{J_{N_{1},j}}{m_{N_{1},j}} \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1},j}^n = \\
2(\Delta \xi)_{N_{1,j}}^{n+1} - (V_{1\eta})_{N_{1} - 2,j}^n + 4(V_{1\eta})_{N_{1} - 1,j}^n - 3(V_{1\eta})_{N_{1},j}^n
\end{align*}

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Two-Point Tangential Velocity Gradient in Normal Direction, \( \nabla V_{\epsilon} \cdot \hat{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using two-point one-sided differencing,
\begin{align*}
- \frac{J_{1,j}}{m_{1,j}} & \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{1,j}^n + \\
\frac{J_{2,j}}{m_{2,j}} & \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{2,j}^n = \\
\frac{\Delta \xi}{m_{1,j}} & \left[ f_{1,j}^{n+1} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{1,j}}{m_{1,j}} \delta_\eta (V_{1\eta})_{1,j}^n \right] + (V_{1\eta})_{1,j}^n - (V_{2\eta})_{2,j}^n
\end{align*}

where \( \delta_\eta \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,
\begin{align*}
- \frac{J_{N_{1} - 1,j}}{m_{N_{1} - 1,j}} & \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1} - 1,j}^n + \\
\frac{J_{N_{1},j}}{m_{N_{1},j}} & \left[ \frac{\xi_y - \xi_x}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\hat{\rho} u) + \frac{\xi_x}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1},j}^n = \\
\frac{\Delta \xi}{m_{N_{1},j}} & \left[ f_{N_{1},j}^{n+1} - \frac{(\xi_x \eta_x + \xi_y \eta_y)_{N_{1},j}}{m_{N_{1},j}} \delta_\eta (V_{1\eta})_{N_{1},j}^n \right] + (V_{1\eta})_{N_{1} - 1,j}^n - (V_{2\eta})_{N_{1},j}^n
\end{align*}
Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point Tangential Velocity Gradient in Normal Direction, \( \nabla V_t \ast \hat{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[
-3 \frac{J_{1,j}}{m_{1,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{1,j}^n + \\
4 \frac{J_{2,j}}{m_{2,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{2,j}^n - \\
3 \frac{J_{3,j}}{m_{3,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{3,j}^n = \\
\frac{2\Delta \xi}{m_{1,j}} \left[ f^n_{1,j} - \frac{(\xi_x u_x + \xi_y u_y)_{1,j}}{m_{1,j}} \delta^n_{\eta(V_t)_{1,j}} \right] + 3(V_t)_{1,j}^n - 4(V_t)_{2,j}^n + (V_t)_{3,j}^n
\]

where \( \delta_n \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
\frac{J_{N_1 - 2,j}}{m_{N_1 - 2,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{N_1 - 2,j}^n - \\
4 \frac{J_{N_1 - 1,j}}{m_{N_1 - 1,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{N_1 - 1,j}^n + \\
3 \frac{J_{N_1,j}}{m_{N_1,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{N_1,j}^n = \\
\frac{2\Delta \xi}{m_{N_1,j}} \left[ f^n_{N_1,j} - \frac{(\xi_x u_x + \xi_y u_y)_{N_1,j}}{m_{N_1,j}} \delta^n_{\eta(V_t)_{N_1,j}} \right] - (V_t)_{N_1 - 2,j}^n + 4(V_t)_{N_1 - 1,j}^n - 3(V_t)_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Linear Extrapolation of Tangential Velocity**

Applying equation (6.14) of Volume 1 at the \( \xi = 0 \) boundary,

\[
\frac{J_{1,j}}{m_{1,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{1,j}^n - \\
\frac{2J_{2,j}}{m_{2,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{2,j}^n + \\
\frac{J_{3,j}}{m_{3,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{3,j}^n = \\
-(V_t)_{1,j}^n + 2(V_t)_{2,j}^n - (V_t)_{3,j}^n
\]

At the \( \xi = 1 \) boundary,
\[
\frac{J_{N_1-2,j}}{m_{N_1-2,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{N_1-2,j}^n - \\
2 \frac{J_{N_1-1,j}}{m_{N_1-1,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{N_1-1,j}^n + \\
\frac{J_{N_1,j}}{m_{N_1,j}} \left[ \frac{\xi_y u - \xi_x v}{\rho} \Delta \hat{\rho} - \frac{\xi_y}{\rho} \Delta (\rho u) + \frac{\xi_x}{\rho} \Delta (\rho v) \right]_{N_1,j}^n = \\
-(V_i)_{N_1-2,i}^n + 2(V_i)_{N_1-1,i}^n - (V_i)_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent normal or tangential velocity boundary condition is specified.
Subroutine BCVVEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NWD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCGRAD</td>
<td>Compute ( y ) or ( r )-velocity boundary conditions.</td>
</tr>
<tr>
<td>BCMET</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- **DEL**
  - Computational grid spacing in sweep direction.
- **IBASE, ISTEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IBC, FBC**
  - Mean flow boundary condition types and values for current sweep direction, specified as IBC(I,J) and FBC(I,J), where I runs from 1 to \( N_q \), corresponding to the \( N_q \) conditions needed, and J = 1 or 2, corresponding to the lower and upper boundaries.
- **IBOUND**
  - Flag specifying boundary; 1 for lower boundary, 2 for upper boundary.
- **IEQ**
  - Boundary condition equation number.
- **IMIN, IMAX**
  - Minimum and maximum indices in the sweep direction.
- **ISWEEP**
  - Current ADI sweep number.
- **IV**
  - Index in the “vectorized” direction, \( i \).
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, \( J^{-1} \).
- **NOUT**
  - Unit number for standard output.
- **NR, NRU, NRV**
  - Array indices associated with the dependent variables \( \rho, \rho u, \) and \( \rho v \).
- **NVD, NPTSD**
  - Leading two dimensions for the arrays A, B, C, and S.
- **RHO, U, V**
  - Static density \( \rho \), and velocities \( u \) and \( v \), at time level \( n \).

**Output**

- **A, B, C**
  - Coefficient submatrices A, B, and C at boundary IBOUND (row IEQ only).
- **S**
  - Source term subvector S at boundary IBOUND (element IEQ only).

**Description**

Subroutine BCVVEL computes coefficients and source terms for \( y \) or \( r \)-velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular \( y \) or \( r \)-velocity boundary conditions in Proteus.\(^\text{14}\)

\(^{14}\) In the following description, for the first ADI sweep the dependent variables should have the superscript *-, representing the intermediate solution, and for the second ADI sweep they should have the superscript \( n \), representing the final solution. For simplicity, however, only the superscript \( n \) is used. The superscripts on all other variables are correct as written.
No Change From Initial Conditions, $\Delta v = 0$

Applying equation (6.3) of Volume 1, and noting that $\frac{\partial g}{\partial \hat{Q}} = \dot{J} \frac{\partial g}{\partial Q}$, we get simply

$$J_{i,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{i,j}^n = 0$$

Specified $y$ or $r$-Velocity, $v = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{i,j}^n = f_{i,j}^{n+1} - v_{i,j}^n$$

Specified Two-Point $y$ or $r$-Velocity Gradient in Coordinate Direction, $\frac{\partial v}{\partial \xi} = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing.

$$- J_{1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{1,j}^n + J_{2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{2,j}^n =

(\Delta \xi) f_{1,j}^{n+1} + v_{1,j}^n - v_{2,j}^n$$

At the $\xi = 1$ boundary,

$$- J_{N_1-1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{N_1-1,j}^n + J_{N_1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{N_1,j}^n =

(\Delta \xi) f_{N_1,j}^{n+1} + v_{N_1-1,j}^n - v_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

Specified Three-Point $y$ or $r$-Velocity Gradient in Coordinate Direction, $\frac{\partial v}{\partial \phi} = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$- 3 J_{1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{1,j}^n + 4 J_{2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{2,j}^n -

J_{3,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{3,j}^n = 2(\Delta \xi) f_{1,j}^{n+1} + 3 v_{1,j}^n - 3 v_{2,j}^n + v_{3,j}^n$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{N_1-2,j}^n - 4 J_{N_1-1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{N_1-1,j}^n +

3 J_{N_1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} \hat{v}) \right]_{N_1,j}^n = 2(\Delta \xi) f_{N_1,j}^{n+1} - v_{N_1-2,j}^n + 3 v_{N_1-1,j}^n - 3 v_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.
**Specified Two-Point \( y \) or \( r \)-Velocity Gradient in Normal Direction, \( \nabla v \cdot \vec{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using two-point one-sided differencing,

\[
-J_{1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{1,j}^n + J_{2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{2,j}^n =
\]

\[
\frac{\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \frac{(\xi_x n_x + \xi_y n_y)_{1,j}}{m_{1,j}} \delta_{\eta} n_{1,j}^n \right] + v_{1,j}^n - v_{2,j}^n
\]

where

\[
m = \sqrt{\xi_x^2 + \xi_y^2}
\]

and \( \delta_{\eta} \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
-J_{N_1 - 1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_1 - 1,j}^n + J_{N_1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_1,j}^n =
\]

\[
\frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \frac{(\xi_x n_x + \xi_y n_y)_{N_1,j}}{m_{N_1,j}} \delta_{\eta} n_{N_1,j}^n \right] + v_{N_1 - 1,j}^n - v_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Three-Point \( y \) or \( r \)-Velocity Gradient in Normal Direction, \( \nabla v \cdot \vec{n} = f \)**

Applying equation (6.12a) of Volume 1 at the \( \xi = 0 \) boundary, and using three-point one-sided differencing,

\[
-3J_{1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{1,j}^n + 4J_{2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{2,j}^n -
\]

\[
J_{3,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{3,j}^n = \frac{2 \Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \frac{(\xi_x n_x + \xi_y n_y)_{1,j}}{m_{1,j}} \delta_{\eta} n_{1,j}^n \right] +
\]

\[
3v_{1,j}^n - 4v_{2,j}^n + v_{3,j}^n
\]

where

\[
m = \sqrt{\xi_x^2 + \xi_y^2}
\]

and \( \delta_{\eta} \) is the centered difference operator presented in Section 5.0 of Volume 1. At the \( \xi = 1 \) boundary,

\[
J_{N_1 - 2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_1 - 2,j}^n - 4J_{N_1 - 1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_1 - 1,j}^n +
\]

\[
3J_{N_1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_1,j}^n = \frac{2 \Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \frac{(\xi_x n_x + \xi_y n_y)_{N_1,j}}{m_{N_1,j}} \delta_{\eta} n_{N_1,j}^n \right] -
\]

\[
v_{N_1 - 2,j}^n - 4v_{N_1 - 1,j}^n + v_{N_1,j}^n
\]

Analogous equations can easily be written for the \( \eta \) boundaries.
**Linear Extrapolation of \( y \) or \( r \)-Velocity**

Applying equation (6.14) of Volume 1 at the \( \zeta = 0 \) boundary,

\[
J_{1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{1,j}^{n} - 2J_{2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{2,j}^{n} + \\
J_{3,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{3,j}^{n} = -v_{1,j}^{n} + 2v_{2,j}^{n} - v_{3,j}^{n}
\]

At the \( \zeta = 1 \) boundary,

\[
J_{N_{1}-2,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1}-2,j}^{n} - 2J_{N_{1}-1,j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1}-1,j}^{n} + \\
J_{N_{1},j} \left[ -\frac{v}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} v) \right]_{N_{1},j}^{n} = -v_{N_{1}-2,j}^{n} + 2v_{N_{1}-1,j}^{n} - v_{N_{1},j}^{n}
\]

Analogous equations can easily be written for the \( \eta \) boundaries.

**Specified Flow Angle, \( \tan^{-1}(v/u) = f \)**

This boundary condition can be rewritten as

\[
\frac{v}{u} = \tan f
\]

where \( f \) is the specified flow angle. Multiplying by \( \rho u \),

\[
(\tan f)\rho u - \rho v = 0
\]

Applying equation (6.5) of Volume 1 to the above equation, we get

\[
J_{i,j} \left[ (\tan f)_{i,j}^{n} + 1\Delta (\hat{\rho} u)_{i,j}^{n} - \Delta (\hat{\rho} v)_{i,j}^{n} \right] = -\left( (\tan f)_{i,j}^{n} + (\rho v)_{i,j}^{n} \right)
\]

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent \( y \)-velocity boundary condition is specified.
Subroutine BCWVEL (IBC,FBC,IEQ,IMIN,IMAX,IBOUND,A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td>BCGRAD BCMET</td>
<td>Compute swirl velocity boundary conditions.</td>
</tr>
</tbody>
</table>

**Input**

- DEL
- IBASE, ISTEP
- IBC, FBC
- IBOUND
- IEQ
- IMIN, IMAX
- ISWEEP
- IV
- JI
- * NOUT
- NR, NRU, NRW
- NVD, NPTSD
- RHO, U, W

**Output**

- A, B, C
- S

**Description**

Subroutine BCWVEL computes coefficients and source terms for swirl velocity boundary conditions. The linearized equations for the various general types of boundary conditions are developed in Section 6.0 of Volume 1. The following sections apply these generalized equations to the particular swirl velocity boundary conditions in *Proteus*.15

In the following description, for the first ADI sweep the dependent variables should have the superscript *, representing the intermediate solution, and for the second ADI sweep they should have the superscript n, representing the final solution. For simplicity, however, only the superscript n is used. The superscripts on all other variables are correct as written.
No Change From Initial Conditions, $\Delta w = 0$

Applying equation (6.3) of Volume 1, and noting that $\partial g/\partial Q = J \partial g/\partial Q$, we get simply

$$J_{i,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{i,j}^n = 0$$

Specified Swirl Velocity, $w = f$

Applying equation (6.5) of Volume 1,

$$J_{i,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{i,j}^n = f_{i,j}^{n+1} - w_{i,j}^n$$

Specified Two-Point Swirl Velocity Gradient in Coordinate Direction, $\partial w/\partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$-J_{1,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{1,j}^n + J_{2,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{2,j}^n = (\Delta \xi) f_{1,j}^{n+1} + w_{1,j}^n - w_{2,j}^n$$

At the $\xi = 1$ boundary,

$$-J_{N_1-1,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1-1,j}^n + J_{N_1,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1,j}^n = (\Delta \xi) f_{N_1,j}^{n+1} + w_{N_1-1,j}^n - w_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

Specified Three-Point Swirl Velocity Gradient in Coordinate Direction, $\partial w/\partial \phi = f$

Applying equation (6.8) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$-3J_{1,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{1,j}^n + 4J_{2,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{2,j}^n - J_{3,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{3,j}^n = 2(\Delta \xi) f_{1,j}^{n+1} + 3w_{1,j}^n - 4w_{2,j}^n + w_{3,j}^n$$

At the $\xi = 1$ boundary,

$$J_{N_1-2,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1-2,j}^n - 4J_{N_1-1,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1-1,j}^n + 3J_{N_1,j} \left[ -\frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1,j}^n = 2(\Delta \xi) f_{N_1,j}^{n+1} - w_{N_1-2,j}^n + 4w_{N_1-1,j}^n - 3w_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.
Specified Two-Point Swirl Velocity Gradient in Normal Direction, $\mathbf{V}_w \cdot \mathbf{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using two-point one-sided differencing,

$$- J_{1,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{1,j}^n + J_{2,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{2,j}^n =$$

$$\frac{\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \left( \frac{\hat{\xi}_x \eta_x + \hat{\xi}_y \eta_y}{m_{1,j}} \right) \delta \eta w_{1,j}^n \right] + w_{1,j}^n - w_{2,j}^n$$

where

$$m = \sqrt{\hat{\xi}_x^2 + \hat{\xi}_y^2}$$

and $\delta \eta$ is the centered difference operator presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$- J_{N_1-1,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1-1,j}^n + J_{N_1,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1,j}^n =$$

$$\frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \left( \frac{\hat{\xi}_x \eta_x + \hat{\xi}_y \eta_y}{m_{N_1,j}} \right) \delta \eta w_{N_1,j}^n \right] + w_{N_1-1,j}^n - w_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

Specified Three-Point Swirl Velocity Gradient in Normal Direction, $\mathbf{V}_w \cdot \mathbf{n} = f$

Applying equation (6.12a) of Volume 1 at the $\xi = 0$ boundary, and using three-point one-sided differencing,

$$- 3 J_{1,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{1,j}^n + 4 J_{2,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{2,j}^n -$$

$$J_{3,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{3,j}^n = 2 \frac{\Delta \xi}{m_{1,j}} \left[ f_{1,j}^{n+1} - \left( \frac{\hat{\xi}_x \eta_x + \hat{\xi}_y \eta_y}{m_{1,j}} \right) \delta \eta w_{1,j}^n \right] +$$

$$3 w_{1,j}^n - 4 w_{2,j}^n + w_{3,j}^n$$

where

$$m = \sqrt{\hat{\xi}_x^2 + \hat{\xi}_y^2}$$

and $\delta \eta$ is the centered difference operator presented in Section 5.0 of Volume 1. At the $\xi = 1$ boundary,

$$J_{N_1-2,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1-2,j}^n - 4 J_{N_1-1,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1-1,j}^n +$$

$$3 J_{N_1,j} \left[ - \frac{w}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta (\hat{\rho} w) \right]_{N_1,j}^n = 2 \frac{\Delta \xi}{m_{N_1,j}} \left[ f_{N_1,j}^{n+1} - \left( \frac{\hat{\xi}_x \eta_x + \hat{\xi}_y \eta_y}{m_{N_1,j}} \right) \delta \eta w_{N_1,j}^n \right] -$$

$$w_{N_1-2,j}^n + 4 w_{N_1-1,j}^n + 3 w_{N_1,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

112 4.0 Proteus Subprograms: BCWVEL Proteus 2-D Programmer's Reference
Linear Extrapolation of Swirl Velocity

Applying equation (6.14) of Volume 1 at the $\xi = 0$ boundary,

$$J_{1,j}\left[-\frac{\omega}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta(\hat{\rho}w)\right]_{1,j}^n - 2J_{2,j}\left[-\frac{\omega}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta(\hat{\rho}w)\right]_{2,j}^n +$$

$$J_{3,j}\left[-\frac{\omega}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta(\hat{\rho}w)\right]_{3,j}^n = -w_{1,i,j}^n + 2w_{2,i,j}^n - w_{3,i,j}^n$$

At the $\xi = 1$ boundary,

$$J_{N_1 - 2,j}\left[-\frac{\omega}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta(\hat{\rho}w)\right]_{N_1 - 2,j}^n - 2J_{N_1 - 1,j}\left[-\frac{\omega}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta(\hat{\rho}w)\right]_{N_1 - 1,j}^n +$$

$$J_{N_1,j}\left[-\frac{\omega}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta(\hat{\rho}w)\right]_{N_1,j}^n = -w_{N_1 - 2,i,j}^n + 2w_{N_1 - 1,i,j}^n - w_{N_1,i,j}^n$$

Analogous equations can easily be written for the $\eta$ boundaries.

Specified Flow Angle, $\tan^{-1}(w/u) = f$

This boundary condition can be rewritten as

$$\frac{w}{u} = \tan f$$

where $f$ is the specified flow angle. Multiplying by $\rho u$,

$$(\tan f)\rho u - \rho w = 0$$

Applying equation (6.5) of Volume 1 to the above equation, we get

$$J_{i,j}\left[(\tan f)_{i,j}^n + \frac{1}{\rho} \Delta(\hat{\rho}u)_{i,j}^n - \Delta(\hat{\rho}w)_{i,j}^n\right] = -\frac{\omega}{\rho} \Delta \hat{\rho} + \frac{1}{\rho} \Delta(\hat{\rho}w)_{i,j}^n + (\rho w)_{i,j}^n$$

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. An error message is generated and execution is stopped if a non-existent swirl velocity boundary condition is specified.
## Subroutine BLIN1

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>TURBBL</td>
<td>ISRACHEQ</td>
<td>Compute inner layer turbulent viscosity along constant $\xi$ lines.</td>
</tr>
</tbody>
</table>

### Input

- **APLUS**
- **CB**
- **CNL**
- **CVK**
- **IDEBUG**
- **ILDAMP**
- **INNER**
- **IPRT1A, IPRT2A**
- **IT**
- **IWALL2**
- **I**
- **LWALL2**
- **MU**
- **MUT**
- **NOUT**
- **NPRT1, NPRT2**
- **N2**
- **RER**
- **RHO, U, V, W**
- **VORT**
- **X, Y**

### Description

Subroutine BLIN1 computes the inner layer turbulent viscosity coefficient ($\mu_i$) at a specified $\xi$ location (i.e., due to walls at $\eta = 0$ and/or $\eta = 1$). Two different inner region models are available - the model of Baldwin and Lomax (1978), and the model of Spalding (1961) and Kleinstein (1967). These are described in Section 9.1.2 of Volume 1.
If both $\eta$ boundaries are solid walls, $(\mu_i)_{\text{inner}}$ is computed separately for each wall, and it is assumed that the two inner regions do not overlap. For each wall, the computation is done inside a loop starting at the wall and moving outward. Once the inner region value exceeds the outer region value, the loop is exited. Thus $\mu_i = (\mu_i)_{\text{inner}}$ until $(\mu_i)_{\text{inner}} > (\mu_i)_{\text{outer}}$, then $\mu_i = (\mu_i)_{\text{outer}}$.

The distribution of $\mu_i$ across the intersection of the inner and outer regions is smoothed using the following formulas. For the $\eta = 0$ wall,

\[
(\mu_i)_j = \frac{1}{4} \left[ (\mu_i)_{j-1} + 2(\mu_i)_{j} + (\mu_i)_{j+1} \right]
\]

\[
(\mu_i)_{j-1} = \frac{1}{4} \left[ (\mu_i)_{j-2} + 2(\mu_i)_{j-1} + (\mu_i)_{j} \right]
\]

where the boundary between the inner and outer regions falls between between $j = j_b - 1$ and $j = j_b$. It should be noted that the unsmoothed value of $(\mu_i)_{j_b}$ is used in the second smoothing formula, not the smoothed value from the first formula. Similarly, for the $\eta = 1$ wall,

\[
(\mu_i)_b = \frac{1}{4} \left[ (\mu_i)_b + 2(\mu_i)_{b+1} + (\mu_i)_b - 1 \right]
\]

\[
(\mu_i)_{b+1} = \frac{1}{4} \left[ (\mu_i)_b + 2(\mu_i)_{b+1} + (\mu_i)_b + 1 \right]
\]

where the boundary between the inner and outer regions falls between between $j = j_b + 1$ and $j = j_b$.

**Remarks**

1. To avoid the possibility of floating point errors, the value of $|\bar{\Omega}|$, used to compute $\tau^+$ and $\eta^+$ is set to a minimum of $10^{-10}$.
2. The Cray search routine ISRCHEQ is used in determining the grid locations for debug printout.
3. This subroutine generates output for the IDEBUG(8) option.
Subroutine BLIN2

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>TURBBL</td>
<td>ISRCHEQ</td>
<td>Compute inner layer turbulent viscosity along constant ( \eta ) lines.</td>
</tr>
</tbody>
</table>

**Input**

- **APLUS**
  - Van Driest damping constant \( A^+ \).
- **CB**
  - Constant \( B \) in the Spalding-Kleinstein inner layer model.
- **CNL**
  - Exponent \( n \) in the Launder-Priddin modified mixing length formula for the inner region of the Baldwin-Lomax turbulence model.
- **CVK**
  - Von Karman mixing length constant used in the inner region of the Baldwin-Lomax and Spalding-Kleinstein models.
- **DUMMY**
  - Outer layer turbulent viscosity coefficient \( (\mu)_{\text{outer}} \) along constant \( \eta \) lines.
- **IDEBUG**
  - Debug flags.
- **ILDAMP**
  - Flag for Launder-Priddin modified mixing length formula in the Baldwin-Lomax inner region model.
- **INNER**
  - Flag for type of inner region model.
- **IPRT1A, IPRT2A**
  - Indices for printout in the \( \xi \) and \( \eta \) directions.
- **IT**
  - Current time step number \( n \).
- **IWALL1**
  - Grid index \( j \) in the \( \xi \) direction.
- **I2**
  - Grid index \( j \) in the \( \eta \) direction.
- **LWALL1**
  - Flags specifying wall locations for \( \xi \) boundaries.
- **MU**
  - Laminar coefficient of viscosity \( \mu_l \).
- **NOUT**
  - Unit number for standard output.
- **NPRT1, NPRT2**
  - Total number of indices for printout in the \( \xi \) and \( \eta \) directions.
- **N1**
  - Number of grid points \( N_1 \) in the \( \xi \) direction.
- **RER**
  - Reference Reynolds number \( Re_r \).
- **RHO, U, V, W**
  - Static density \( \rho \), and velocities \( u, v, \) and \( w \).
- **VORT**
  - Total vorticity magnitude.
- **X, Y**
  - Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \).

**Output**

- **DUMMY**
  - Turbulent viscosity coefficient \( \mu_t \) along constant \( \eta \) lines.

**Description**

Subroutine BLIN2 computes the inner layer turbulent viscosity coefficient \( (\mu)_{\text{inner}} \) at a specified \( \eta \) location (i.e., due to walls at \( \xi = 0 \) and/or \( \xi = 1 \)). The procedure is exactly analogous to that used in subroutine BLIN1.
Subroutine BLKOUT (I1PT, I2PT, A, B, C, S, NVD, NPTSD)

Called by | Calls | Purpose
--- | --- | ---
ADI | | Print coefficient blocks at specified indices in the ξ and η directions.
AVISC1 | | 
AVISC2 | | 
BCGEN | | 
FILTER | | 

**Input**

A, B, C

* IHSTAG
  * ISWEEP
  * ISWIRL
  * I1PT, I2PT
  * NC, NXM, NYM, NZM, NEN

NEQ

* NOUT
  * NVD, NPTSD
  * S

**Output**

None.

**Description**

Subroutine BLKOUT prints the coefficient block submatrices A, B, and C, and the source term subvector S at the grid points specified by I1PT and I2PT. This is the routine that actually prints the output for the IDEBUG(1) through IDEBUG(4) options.
Subroutine BLK2 (A, B, C, S, NVD, NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECT</td>
<td></td>
<td>Solve 2 x 2 block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- A, B, C: Coefficient submatrices A, B, and C
- NPTS: Number of grid points in the sweep direction, N.
- NV: Number of grid points in the "vectorized" direction, N\_.
- NVD, NPTSD: Leading two dimensions for the arrays A, B, C, and S.
- S: Source term subvector.

**Output**

- S: Computed solution subvector.

**Description**

Subroutine BLK2 solves a block tridiagonal system of equations with 2 x 2 blocks using the block matrix version of the Thomas algorithm. The algorithm is described in Section 7.2.1 of Volume 1. For clarity, that description involves additional "new" matrices D, E, and \( \Delta \hat{Q}' \). In Fortran, however, we can save storage by overwriting B, C, and S. The following table relates the algorithm as implemented in Fortran to the notation used in Volume 1, for the first ADI sweep. An exactly analogous procedure is followed for the second sweep.

<table>
<thead>
<tr>
<th>Step</th>
<th>In Fortran</th>
<th>In Volume 1 Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D_2 = B_2</td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>Solve B_2E_2 = C_1 for E_2, storing result in C_2</td>
<td>E_2 = D_1^{-1}C_2</td>
</tr>
<tr>
<td>2b</td>
<td>Solve B_2( \Delta \hat{Q}' ) = S_2 for ( \Delta \hat{Q}' ), storing result in S_2</td>
<td>( \Delta \hat{Q}'_2 = D_1^{-1}S_2 )</td>
</tr>
<tr>
<td>3a</td>
<td>For ( i = 3 ) to ( N_1 - 1 ), Compute B_i - A,C_{i-1}, storing result in B_i</td>
<td>B_i = B_i - A,E_{i-1}</td>
</tr>
<tr>
<td>3b</td>
<td>Compute S_i - A,S_{i-1}, storing result in S_i</td>
<td>S_i = A( \hat{\Delta \hat{Q}'}_{i-1} )</td>
</tr>
<tr>
<td>3c</td>
<td>Solve B_iE_i = C_i for E_i, storing result in C_i</td>
<td>E_i = D_i^{-1}C_i</td>
</tr>
<tr>
<td>3d</td>
<td>Solve B_i( \Delta \hat{Q}' ) = S, for ( \Delta \hat{Q}' ), storing result in S_i</td>
<td>( \Delta \hat{Q}' = D_i^{-1}(S_i - A( \Delta \hat{Q}'_{i-1} )) )</td>
</tr>
<tr>
<td>4</td>
<td>( \Delta \hat{Q}'<em>{N_1-1} = \Delta \hat{Q}'</em>{N_1-1} )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>For ( i = N_1 - 2 ) to 2, Compute S_i - C,S_{i-1}, storing result in S_i</td>
<td>( \Delta \hat{Q}'<em>i = \Delta \hat{Q}' - E_i\hat{\Delta \hat{Q}'}</em>{i-1} )</td>
</tr>
</tbody>
</table>

**Remarks**

1. The notation used in the comments in BLK2 is consistent with the notation used in the description of the algorithm in Volume 1. However, BLK2 is actually used to solve the \( k-\epsilon \) turbulence model equations, and the boundary conditions for these equations are treated explicitly. That's why the index in BLK2 runs from \( i = 2 \) to \( N_1 - 1 \), instead of from \( i = 1 \) to \( N_1 \). In addition, in BLK2 the matrix \( D^{-1} \) is computed directly, rather than by LU decomposition.
2. The Thomas algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
Subroutine BLK2P (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECT</td>
<td></td>
<td>Solve 2 × 2 periodic block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- A, B, C: Coefficient submatrices A, B, and C
- NPTS: Number of grid points in the sweep direction, N.
- NV: Number of grid points in the "vectorized" direction, Nv.
- NVD, NPTSD: Leading two dimensions for the arrays A, B, C, and S.
- S: Source term subvector S.

**Output**

- S: Computed solution subvector.

**Description**

Subroutine BLK2P solves a periodic block tridiagonal system of equations with 2 × 2 blocks. An efficient algorithm similar to the block matrix version of the Thomas algorithm is used to solve the equations. The algorithm is described in Section 7.2.2 of Volume 1. For clarity, that description involves additional "new" matrices D, E, F, G, and ΔQ'. In Fortran, however, we can save storage by overwriting A, B, C, and S. The following table relates the algorithm as implemented in Fortran to the notation used in Volume 1, for the first ADI sweep. An exactly analogous procedure is followed for the second sweep.

<table>
<thead>
<tr>
<th>Step</th>
<th>In Fortran</th>
<th>In Volume 1 Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td></td>
<td>D_2 = B_2</td>
</tr>
<tr>
<td>1b</td>
<td></td>
<td>F_2 = C_{N_1}</td>
</tr>
<tr>
<td>2a</td>
<td>Solve B_2 E_2 = C_2 for E_2, storing result in C_2</td>
<td>E_2 = D_2 C_{N_1}</td>
</tr>
<tr>
<td>2b</td>
<td>Solve B_2 G_2 = A_2 for G_2, storing result in A_2</td>
<td>G_2 = D_2 A_{N_1}</td>
</tr>
<tr>
<td>2c</td>
<td>Solve B_2 ΔQ'_2 = S_2 for ΔQ'_2, storing result in S_2</td>
<td>ΔΩ'<em>2 = D_2 S</em>{N_1}</td>
</tr>
</tbody>
</table>

For i = 3 to N_1 - 1,

- 3a: Compute B_i - A_i C_{i-1}, storing result in B_i
- 3b: Compute S_i - A_i S_{i-1}, storing result in S_i
- 3c: Compute - A_i A_{i-1}, storing result in A_i
- 3d: Solve B_i E_i = C_i, for E_i, storing result in C_i
- 3e: Solve B_i G_i = A_i, for G_i, storing result in A_i
- 3f: Solve B_i ΔQ'_i = S_i for ΔQ'_i, storing result in S_i
- 3g: Compute B_{N_1} - C_{N_1} A_{N_1}, storing result in B_{N_1}
- 3h: Compute S_{N_1} - C_{N_1} S_{N_1}, storing result in S_{N_1}
- 3i: Compute - C_{N_1} C_{N_1}, storing result in C_{N_1}
<table>
<thead>
<tr>
<th>Step</th>
<th>In Fortran</th>
<th>In Volume 1 Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4a</td>
<td>Compute $A_{N1-1} + C_{N1-1}$, storing result in $A_{N1-1}$</td>
<td>$G_{N1-1} = D_{N1-1}^{-1}(C_{N1-1} - A_{N1-1}G_{N1-2})$</td>
</tr>
<tr>
<td>4b</td>
<td>Compute $A_{N1} + C_{N1}$, storing result in $C_{N1}$</td>
<td>$F_{N1-1} = A_{N1} - F_{N1-2}E_{N1-2}$</td>
</tr>
<tr>
<td>4c</td>
<td>Compute $B_{N1} - C_{N1}A_{N1-1}$, storing result in $B_{N1}$</td>
<td>$D_{N1} = B_{N1} - \sum_{i=2}^{N1-1} F_iG_i$</td>
</tr>
<tr>
<td>4d</td>
<td>Compute $S_{N1} - C_{N1}S_{N1-1}$, storing result in $S_{N1}$</td>
<td>$S_{N1} = \sum_{i=2}^{N1-1} F_i\hat{Q}'_i$</td>
</tr>
<tr>
<td>4f</td>
<td>Solve $B_{N1}\hat{Q}'<em>{N1} = S</em>{N1}$ for $\hat{Q}'<em>{N1}$, storing result in $S</em>{N1}$</td>
<td>$\hat{Q}'<em>{N1} = D</em>{N1}^{-1}(S_{N1} - \sum_{i=2}^{N1-1} F_i\hat{Q}'_i)$</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>$\hat{Q}<em>{N1} = \hat{Q}'</em>{N1}$</td>
</tr>
<tr>
<td>6</td>
<td>Compute $S_{N1-1} - A_{N1-1}S_{N1}$, storing result in $S_{N1-1}$</td>
<td>$\hat{Q}<em>{N1-1} = \hat{Q}'</em>{N1-1} - G_{N1-1}\hat{Q}_{N1}$</td>
</tr>
<tr>
<td>For i = $N_1 - 2$ to 2,</td>
<td></td>
<td>$\hat{Q}<em>i = \hat{Q}'<em>i - G_i\hat{Q}</em>{N1} - E_i\hat{Q}</em>{i+1}$</td>
</tr>
<tr>
<td>7</td>
<td>Compute $S_i - A_iS_{N1} - C_iS_{i-1}$, storing result in $S_i$</td>
<td>$\hat{Q}_i = \hat{Q}'_i$</td>
</tr>
<tr>
<td>8</td>
<td>Set $S_i = S_{N1}$</td>
<td>$\hat{Q}_i = \hat{Q}'_i$</td>
</tr>
</tbody>
</table>

**Remarks**

1. The notation used in the comments in BLK2P is consistent with the notation used in the description of the algorithm in Volume 1. However, in BLK2P the matrix $D^{-1}$ is computed directly, rather than by LU decomposition.

2. The solution algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or IV, subscript on the $A$, $B$, $C$, and $S$ arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
Subroutine BLK3 (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td>FILTER</td>
<td>Solve 3 x 3 block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- **A, B, C**: Coefficient submatrices A, B, and C
- **NPTS**: Number of grid points in the sweep direction, N.
- **NV**: Number of grid points in the "vectorized" direction, Nv.
- **NVD, NPTSD**: Leading two dimensions for the arrays A, B, C, and S.
- **S**: Source term subvector S.

**Output**

- **S**: Computed solution subvector.

**Description**

Subroutine BLK3 solves a block tridiagonal system of equations with 3 x 3 blocks using the block matrix version of the Thomas algorithm. Subroutine FILTER is called in an attempt to eliminate any zero values on the diagonal of the submatrix B at the two boundaries. These can occur when mean flow boundary conditions are specified using the JBC and/or IBC input parameters, depending on the initial conditions and the order of the boundary conditions.

The algorithm is described in Section 7.2.1 of Volume 1. For clarity, that description involves additional "new" matrices D, E, and \( \hat{Q}' \). In Fortran, however, storage is saved by overwriting B, C, and S. The following table relates the algorithm as implemented in Fortran to the notation used in Volume 1, for the first ADI sweep. An exactly analogous procedure is followed for the second sweep.

<table>
<thead>
<tr>
<th>Step</th>
<th>In Fortran</th>
<th>In Volume 1 Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D_1 = B_1</td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>LU decompose B_1, storing result in B_1</td>
<td>LU decomposition of D_1</td>
</tr>
<tr>
<td>2b</td>
<td>Solve B_1E_1 = C_1 for E_1 using LU decomposition of B_1, storing result in C_1</td>
<td>E_1 = D_1^{-1}C_1</td>
</tr>
<tr>
<td>2c</td>
<td>Solve B_1( \hat{Q}'_1 ) = S_1 for ( \hat{Q}'_1 ) using LU decomposition of B_1, storing result in S_1</td>
<td>( \hat{Q}'_1 = D_1^{-1}S_1 )</td>
</tr>
<tr>
<td></td>
<td>For ( i = 2 ) to ( N_1 ),</td>
<td></td>
</tr>
<tr>
<td>3a</td>
<td>Compute B_1 = A_1 C_1, storing result in B_1</td>
<td>D_i = B_i - A_i E_i^{-1}</td>
</tr>
<tr>
<td>3b</td>
<td>Compute S_1 = A_1 S_1, storing result in S_1</td>
<td>S_i = D_i^{-1}C_i</td>
</tr>
<tr>
<td>3c</td>
<td>LU decompose B_1, storing result in B_1</td>
<td>LU decomposition of D_i</td>
</tr>
<tr>
<td>3d</td>
<td>Solve B_1E_1 = C_1 for E_1 using LU decomposition of B_1, storing result in C_1</td>
<td>E_1 = D_1^{-1}C_1</td>
</tr>
<tr>
<td>3e</td>
<td>Solve B_1( \hat{Q}'_1 ) = S_1 for ( \hat{Q}'_1 ) using LU decomposition of B_1, storing result in S_1</td>
<td>( \hat{Q}'<em>1 = D_1^{-1}(S_i - A_1 \hat{Q}'</em>{i-1}) )</td>
</tr>
<tr>
<td>4</td>
<td>( \hat{Q}'<em>{N_1} = \hat{Q}'</em>{N_1} )</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>In Fortran</td>
<td>In Volume 1 Notation</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>5</td>
<td>For (i = N_t - 1) to 1, Compute (S_i = C_{S,i-1}), storing result in (S_i)</td>
<td>(\Delta \hat{Q}<em>i = \Delta \hat{Q}</em>{i-1} - E \cdot \Delta \hat{Q}_{i-1})</td>
</tr>
</tbody>
</table>

**Remarks**

1. The notation used in the comments in BLK3 is consistent with the notation used in the description of the algorithm in Volume 1.

2. The Thomas algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or IV, subscript on the \(A\), \(B\), \(C\), and \(S\) arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
Subroutine BLK3P (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td></td>
<td>Solve 3 x 3 periodic block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- A, B, C: Coefficient submatrices A, B, and C
- NPTS: Number of grid points in the sweep direction, N.
- NV: Number of grid points in the "vectorized" direction, Nv.
- NVD, NPTSD: Leading two dimensions for the arrays A, B, C, and S.
- S: Source term subvector S.

**Output**

- S: Computed solution subvector.

**Description**

Subroutine BLK3P solves a periodic block tridiagonal system of equations with 3 x 3 blocks. An efficient algorithm similar to the block matrix version of the Thomas algorithm is used to solve the equations. The algorithm is described in Section 7.2.2 of Volume 1. For clarity, that description involves additional "new" matrices D, E, F, G, and ΔQ'. In Fortran, however, storage is saved by overwriting A, B, C, and S. The following table relates the algorithm as implemented in Fortran to the notation used in Volume 1, for the first ADI sweep. An exactly analogous procedure is followed for the second sweep.

<table>
<thead>
<tr>
<th>Step</th>
<th>In Fortran</th>
<th>In Volume 1 Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>LU decompose B₂, storing result in B₂</td>
<td>LU decomposition of D₂</td>
</tr>
<tr>
<td>2b</td>
<td>Solve B₂E₂ = C₂ for E₂ using LU decomposition of B₂, storing result in C₂</td>
<td>E₂ = D₂⁻¹C₂</td>
</tr>
<tr>
<td>2c</td>
<td>Solve B₂G₂ = A₂ for G₂ using LU decomposition of B₂, storing result in A₂</td>
<td>G₂ = D₂⁻¹A₂</td>
</tr>
<tr>
<td>2d</td>
<td>Solve B₂ΔQ'₂ = S₂ for ΔQ'₂ using LU decomposition of B₂, storing result in S₂</td>
<td>ΔQ'₂ = D₂⁻¹S₂</td>
</tr>
</tbody>
</table>

124 4.0 Proteus Subprograms: BLK3P Proteus 2-D Programmer's Reference
<table>
<thead>
<tr>
<th>Step</th>
<th>In Fortran</th>
<th>In Volume 1 Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3a</td>
<td>Compute $B_i - A_i C_{i-1}$, storing result in $B_i$</td>
<td>$D_i = B_i - A_i E_{i-1}$</td>
</tr>
<tr>
<td>3b</td>
<td>Compute $S_i - A_i S_{i-1}$, storing result in $S_i$</td>
<td>$S_i - A_i \Delta \hat{Q}'_{i-1}$</td>
</tr>
<tr>
<td>3c</td>
<td>Compute $-A_i A_{i-1}$, storing result in $A_i$</td>
<td>$-A_i G_{i-1}$</td>
</tr>
<tr>
<td>3d</td>
<td>LU decompose $B_i$, storing result in $B_i$</td>
<td>LU decomposition of $D_i$</td>
</tr>
<tr>
<td>3e</td>
<td>Solve $B_i E_i = C_i$ for $E_i$, using LU decomposition of $B_i$, storing result in $C_i$</td>
<td>$E_i = D_i^{-1} C_i$</td>
</tr>
<tr>
<td>3f</td>
<td>Solve $B_i G_i = A_i$, for $G_i$, using LU decomposition of $B_i$, storing result in $A_i$</td>
<td>$G_i = D_i^{-1} A_i G_{i-1}$</td>
</tr>
<tr>
<td>3g</td>
<td>Solve $B_i \Delta \hat{Q}' = S_i$, for $\Delta \hat{Q}'$ using LU decomposition of $B_i$, storing result in $S_i$</td>
<td>$\Delta \hat{Q}' = D_i^{-1}(S_i - A_i \Delta \hat{Q}'_{i-1})$</td>
</tr>
<tr>
<td>3h</td>
<td>Compute $B_{N_1} - C_{N_1} A_{N_1-1}$, storing result in $B_{N_1}$</td>
<td>$B_{N_1} = \sum_{j=2}^{N_{i-1}} F_i G_j$</td>
</tr>
<tr>
<td>3i</td>
<td>Compute $S_{N_1} - C_{N_1} S_{N_1-1}$, storing result in $S_{N_1}$</td>
<td>$S_{N_1} = \sum_{j=2}^{N_{i-1}} F_i \Delta \hat{Q}'_j$</td>
</tr>
<tr>
<td>3j</td>
<td>Compute $-C_{N_1} C_{N_1-1}$, storing result in $C_{N_1}$</td>
<td>$F_i = -F_{i-1} E_{i-1}$</td>
</tr>
<tr>
<td>4a</td>
<td>Compute $A_{N_1-1} + C_{N_1-1}$, storing result in $A_{N_1-1}$</td>
<td>$G_{N_1-1} = D_{N_1-1} C_{N_1-1} - A_{N_1-1} G_{N_1-2}$</td>
</tr>
<tr>
<td>4b</td>
<td>Compute $A_{N_1} + C_{N_1}$, storing result in $C_{N_1}$</td>
<td>$F_{N_1-1} = A_{N_1} - F_{N_1-2} E_{N_1-2}$</td>
</tr>
<tr>
<td>4c</td>
<td>Compute $B_{N_1} - C_{N_1} A_{N_1-1}$, storing result in $B_{N_1}$</td>
<td>$D_{N_1} = B_{N_1} - \sum_{i=2}^{N_{i-1}} F_i G_i$</td>
</tr>
<tr>
<td>4d</td>
<td>Compute $S_{N_1} - C_{N_1} S_{N_1-1}$, storing result in $S_{N_1}$</td>
<td>$S_{N_1} = \sum_{i=2}^{N_{i-1}} F_i \Delta \hat{Q}'_i$</td>
</tr>
<tr>
<td>4e</td>
<td>LU decompose $B_{N_1}$, storing result in $B_{N_1}$</td>
<td>LU decomposition of $D_{N_1}$</td>
</tr>
<tr>
<td>4f</td>
<td>Solve $B_{N_1} \Delta \hat{Q}'<em>{N_1} = S</em>{N_1}$ for $\Delta \hat{Q}'<em>{N_1}$ using LU decomposition of $B</em>{N_1}$, storing result in $S_{N_1}$</td>
<td>$\Delta \hat{Q}'<em>{N_1} = D</em>{N_1} (S_{N_1} - \sum_{i=2}^{N_{i-1}} F_i \Delta \hat{Q}'_i)$</td>
</tr>
</tbody>
</table>

**Remarks**

1. The notation used in the comments in BLK3P is consistent with the notation used in the description of the algorithm in Volume 1.

2. The solution algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or IV, subscript on the $A$, $B$, $C$, and $S$ arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
Subroutine BLK4 (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td>FILTER</td>
<td>Solve 4 × 4 block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- A, B, C: Coefficient submatrices A, B, and C
- NPTS: Number of grid points in the sweep direction, N.
- NV: Number of grid points in the "vectorized" direction, Nv.
- NVD, NPTSD: Leading two dimensions for the arrays A, B, C, and S.
- S: Source term subvector S.

**Output**

- S: Computed solution subvector.

**Description**

Subroutine BLK4 solves a block tridiagonal system of equations with 4 × 4 blocks using the block matrix version of the Thomas algorithm. Subroutine FILTER is called in an attempt to eliminate any zero values on the diagonal of the submatrix B at the two boundaries. These can occur when mean flow boundary conditions are specified using the JBC and/or IBC input parameters, depending on the initial conditions and the order of the boundary conditions.

The algorithm is described in Section 7.2.1 of Volume 1. For clarity, that description involves additional "new" matrices D, E, and ΔQ. In Fortran, however, storage is saved by overwriting B, C, and S. The algorithm is identical to that used in subroutine BLK3. See the description of that subroutine for a table relating the algorithm as implemented in Fortran to the notation used in Volume 1.

**Remarks**

1. The notation used in the comments in BLK4 is consistent with the notation used in the description of the algorithm in Volume 1.

2. The Thomas algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
Subroutine BLK4P (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td></td>
<td>Solve $4 \times 4$ periodic block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- A, B, C: Coefficient submatrices A, B, and C
- NPTS: Number of grid points in the sweep direction, N.
- NV: Number of grid points in the "vectorized" direction, $N_v$.
- NVD, NPTSD: Leading two dimensions for the arrays A, B, C, and S.
- S: Source term subvector S.

**Output**

- S: Computed solution subvector.

**Description**

Subroutine BLK4P solves a periodic block tridiagonal system of equations with $4 \times 4$ blocks. An efficient algorithm similar to the block matrix version of the Thomas algorithm is used to solve the equations. The algorithm is described in Section 7.2.2 of Volume 1. For clarity, that description involves additional "new" matrices D, E, F, G, and $\Delta Q'$. In Fortran, however, storage is saved by overwriting A, B, C, and S. The algorithm is identical to that used in subroutine BLK3P. See the description of that subroutine for a table relating the algorithm as implemented in Fortran to the notation used in Volume 1.

**Remarks**

1. The notation used in the comments in BLK4P is consistent with the notation used in the description of the algorithm in Volume 1.

2. The solution algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
Subroutine BLK5 (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td>FILTER</td>
<td>Solve 5 × 5 block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- A, B, C: Coefficient submatrices A, B, and C
- NPTS: Number of grid points in the sweep direction, $N$.
- NV: Number of grid points in the "vectorized" direction, $N_v$.
- NVD, NPTSD: Leading two dimensions for the arrays A, B, C, and S.
- S: Source term subvector S.

**Output**

- S: Computed solution subvector.

**Description**

Subroutine BLK5 solves a block tridiagonal system of equations with 5 × 5 blocks using the block matrix version of the Thomas algorithm. Subroutine FILTER is called in an attempt to eliminate any zero values on the diagonal of the submatrix B at the two boundaries. These can occur when mean flow boundary conditions are specified using the JBC and/or IBC input parameters, depending on the initial conditions and the order of the boundary conditions.

The algorithm is described in Section 7.2.1 of Volume 1. For clarity, that description involves additional "new" matrices D, E, and $\Delta \tilde{Q}$. In Fortran, however, storage is saved by overwriting B, C, and S. The algorithm is identical to that used in subroutine BLK3. See the description of that subroutine for a table relating the algorithm as implemented in Fortran to the notation used in Volume 1.

**Remarks**

1. The notation used in the comments in BLK5 is consistent with the notation used in the description of the algorithm in Volume 1.
2. The Thomas algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or IV, subscript on the A, B, C, and S arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
Subroutine BLK5P (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td></td>
<td>Solve $5 \times 5$ periodic block tridiagonal system of equations.</td>
</tr>
</tbody>
</table>

**Input**

- **A, B, C**
  - Coefficient submatrices $A$, $B$, and $C$
- **NPTS**
  - Number of grid points in the sweep direction, $N$.
- **NV**
  - Number of grid points in the "vectorized" direction, $N_v$.
- **NVD, NPTSD**
  - Leading two dimensions for the arrays $A$, $B$, $C$, and $S$.
- **S**
  - Source term subvector $S$.

**Output**

- **S**
  - Computed solution subvector.

**Description**

Subroutine BLK5P solves a periodic block tridiagonal system of equations with $5 \times 5$ blocks. An efficient algorithm similar to the block matrix version of the Thomas algorithm is used to solve the equations. The algorithm is described in Section 7.2.2 of Volume 1. For clarity, that description involves additional "new" matrices $D$, $E$, $F$, $G$, and $\Delta \tilde{Q}'$. In Fortran, however, storage is saved by overwriting $A$, $B$, $C$, and $S$. The algorithm is identical to that used in subroutine BLK3P. See the description of that subroutine for a table relating the algorithm as implemented in Fortran to the notation used in Volume 1.

**Remarks**

1. The notation used in the comments in BLK5P is consistent with the notation used in the description of the algorithm in Volume 1.
2. The solution algorithm is recursive and therefore cannot be vectorized in the sweep direction. In an ADI procedure, however, if the coefficients and source terms are stored in both directions, the algorithm can be vectorized in the non-sweep direction. That is the reason for the first, or $IV$, subscript on the $A$, $B$, $C$, and $S$ arrays. It was added simply to allow vectorization of the BLK routines. This increases the storage required by the program, but greatly decreases the CPU time required for the ADI solution.
### BLOCK DATA Subprogram

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Set default values for input parameters, plus a few other parameters.</td>
</tr>
</tbody>
</table>

### Input

None.

### Output

All namelist input parameters, plus:

- **CCP1, CCP2, CCP3, CCP4**: Constants in formula for specific heat. \((8.53 \times 10^3, 3.12 \times 10^4, 2.065 \times 10^5, 7.83 \times 10^6)\)\(^{16}\)
- **CK1, CK2**: Constants in formula for laminar thermal conductivity coefficient. \((7.4907 \times 10^{-3}, 350.0)\)\(^{16}\)
- **CMU1, CMU2**: Constants in formula for laminar viscosity coefficient. \((7.3035 \times 10^{-7}, 198.6)\)\(^{16}\)
- **GC**: Proportionality factor \(g\) in Newton’s second law. \((32.174)\)\(^{16}\)
- **IBCELM**: Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the \(\xi\) and \(\eta\) directions at either boundary; 0 if elimination is not necessary, 1 if it is. \((2*0, 2*0)\)
- **IBVUP**: Flags for updating boundary values from first sweep after second sweep; 0 if updating is not necessary, 1 if it is. \((0, 0)\)
- **ICONV**: Convergence flag; 1 if converged, 0 if not. \((0)\)
- **IGINT**: Flags for grid interpolation requirement for the \(\xi\) and \(\eta\) directions; 0 if interpolation is not necessary, 1 if it is. \((0, 0)\)
- **ITBEG**: The time level \(n\) at the beginning of a run. \((1)\)
- **KBCPER**: Flags for spatially periodic boundary conditions in the \(\xi\) and \(\eta\) directions; 0 for non-periodic, 1 for periodic. \((0, 0)\)
- **NC, NXM, NYM, NZM, NEN**: Array indices associated with the continuity, \(x\)-momentum, \(y\)-momentum (or \(r\)-momentum if axisymmetric), swirl momentum, and energy equations. \((1,2,3,5,4)\)
- **NIN**: Unit number for standard input. \((5)\)
- **NR, NRU, NRV, NRW, NET**: Array indices associated with the dependent variables \(\rho, \rho u, \rho v, \rho w\), and \(E_r\). \((1,2,3,5,4)\)
- **RAX**: 1 for two-dimensional planar flow, and the local radius \(r\) for axisymmetric flow. \((NMAXP*1.0)\)
- **TAU**: Initial time value \(\tau\). \((NTOTP*0.0)\)

\(^{16}\) These values are for reference conditions specified in English units. Values for SI units are set in subroutine INPUT.
Description

The BLOCK DATA routine is used to set default values for all the input parameters, plus various other parameters and constants. The defaults for all the input parameters are given as part of the standard input description in Section 3.1 of Volume 2. The values for the other parameters and constants set in BLOCK DATA are given in parentheses in the above output description. Note that some of these values assume English units are being used to specify reference conditions. If SI units are being used, these values are re-defined in subroutine INPUT.

Remarks

1. Most of the default values are defined directly, but some, like the reference viscosity MUR, are set equal to zero and defined in subroutine INPUT if not specified by the user.
Subroutine BLOUT1

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>TURBBL</td>
<td>ISAMAX</td>
<td>Compute outer layer turbulent viscosity, using the algebraic Baldwin-</td>
</tr>
<tr>
<td></td>
<td>ISAMIN</td>
<td>Lomax model, along constant $\xi$ lines.</td>
</tr>
<tr>
<td></td>
<td>ISRCHEQ</td>
<td></td>
</tr>
</tbody>
</table>

### Input

- **APLUS**
  - Van Driest damping constant $A^*$.
- **CB**
  - Constant $B$ in the Klebanoff intermittency factor.
- **CCLAU**
  - Clauser constant $K$ in the Baldwin-Lomax outer region model.
- **CCP**
  - Constant $C_o$ in the Baldwin-Lomax outer region model.
- **CKLEB, CKMIN**
  - Constants $C_{Kleb}$ and $(C_{Kleb})_{min}$ in the Klebanoff intermittency factor.
- **CNA**
  - Exponent $n$ in the formula used to average the two outer region $\mu$, profiles that result when both boundaries in a coordinate direction are solid surfaces.
- **CWK**
  - Constant $C_{ww}$ in the Baldwin-Lomax outer region model.
- **IDEBUG**
  - Debug flags.
- **IPRT1A, IPRT2A**
  - Indices for printout in the $\xi$ and $\eta$ directions.
- **IT**
  - Current time step number $n$.
- **IWALL2**
  - Flags indicating whether or not the $\eta$ boundaries are walls.
- **I**
  - Grid index $i$ in the $\xi$ direction.
- **LWALL2**
  - Flags specifying wall locations for $\eta$ boundaries.
- **MU**
  - Laminar coefficient of viscosity $\mu_i$.
- **NOUT**
  - Unit number for standard output.
- **NPRT1, NPRT2**
  - Total number of indices for printout in the $\xi$ and $\eta$ directions.
- **N2**
  - Number of grid points $N_2$ in the $\eta$ direction.
- **RER**
  - Reference Reynolds number $Re$.
- **RHO, U, V, W**
  - Static density $\rho$, and velocities $u$, $v$, and $w$.
- **VORT**
  - Total vorticity magnitude.
- **X, Y**
  - Cartesian coordinates $x$ and $y$, or cylindrical coordinates $x$ and $r$.

### Output

- **LWAKE2**
  - Grid index $j$ in the $\eta$ direction used as the origin for computing length scales for free turbulent flows.
- **MUT**
  - Outer layer turbulent viscosity coefficient $(\mu_i)_{outer}$ along constant $\xi$ lines.

### Description

Subroutine BLOUT1 computes the outer layer turbulent viscosity coefficient $(\mu_i)_{outer}$ at a specified $\xi$ location (i.e., due to walls at $\eta = 0$ and/or $\eta = 1$, or due to a free turbulent flow in the $\xi$ direction) using the
algebraic eddy viscosity model of Baldwin and Lomax (1978). The model is described in Section 9.1 of Volume 1. The steps performed in BLOUT1 are as follows:

1. Find the values and locations of \( |\vec{V}|_{\text{max}} \) and \( |\vec{V}|_{\text{min}} \).

2. Compute the parameter

\[
F_{\text{wake}} = \begin{cases} 
    y_{\text{max}} F_{\text{max}} & \text{for wall-bounded flows} \\
    C_{\text{wk}} \frac{\nu}{\text{diff}} \frac{y_{\text{max}}}{F_{\text{max}}} & \text{for free turbulent flows}
\end{cases}
\]

where \( C_{\text{wk}} \) is a constant taken as 0.25, and

\[
V_{\text{diff}} = |\vec{V}|_{\text{max}} - |\vec{V}|_{\text{min}}
\]

where \( \vec{V} \) is the total velocity vector. The procedure for computing the parameter \( F_{\text{max}} \) depends on whether a solid wall exists at \( \eta = 0 \) and/or at \( \eta = 1 \).

3. If, at the current \( \xi \) location, the \( \eta = 0 \) boundary is a solid wall, compute \( (F_{\text{max}})_1 \) as follows:

3a. For \( \eta \)-indices from the wall to the location of \( |\vec{V}|_{\text{max}} \), compute

\[
F(y_{\eta}) = y_{\eta} |\vec{V}|_{\text{max}} \left( 1 - e^{-y_{\eta}/\Delta r} \right)
\]

3b. Get the location of the maximum value of \( F(y_{\eta}) \), calling its \( \eta \)-index \( L_1 \). Tentatively set \( (L_{\text{max}})_1 = L_1 \).

3c. Search outward from this location for a local minimum in \( F(y_{\eta}) \). If one is found, call its \( \eta \)-index \( L_{\text{min}} \).

3d. If a local minimum exists, get the location of the next maximum value of \( F(y_{\eta}) \), calling its \( \eta \)-index \( L_2 \). This is the location of the second peak in \( F(y_{\eta}) \). Let

\[
\Delta F_1 = F(y_{\eta})_{L_1} - F(y_{\eta})_{L_{\text{min}}}
\]

\[
\Delta F_2 = F(y_{\eta})_{L_2} - F(y_{\eta})_{L_{\text{min}}}
\]

Then, if \( \Delta F_2 > 0.25 \Delta F_1 \), set \( (L_{\text{max}})_1 = L_2 \). This test is intended to filter false peaks resulting from noise, or wiggles, in \( F(y_{\eta}) \).

3e. Set \( (F_{\text{max}})_1 = F(y_{\eta}) \) at the \( \eta \)-index \( (L_{\text{max}})_1 \), and set \( y_{\text{max}} \) equal to the corresponding value of \( y_{\eta} \).

4. If, at the current \( \xi \) location, the \( \eta = 1 \) boundary is a solid wall, compute \( (F_{\text{max}})_2 \). The procedure is exactly analogous to the procedure described in steps 3a-3e for computing \( (F_{\text{max}})_1 \), at the \( \eta = 0 \) wall.

5. If neither \( \eta \) boundary is a solid wall, a free turbulent flow in the \( \xi \) direction is assumed. In this case, the required value of \( F_{\text{max}} \) is computed as follows:

5a. For \( \eta \)-indices between the locations of \( |\vec{V}|_{\text{min}} \) and \( |\vec{V}|_{\text{max}} \), and using the location of \( |\vec{V}|_{\text{max}} \) as the origin for \( y_{\eta} \), compute

\[
F(y_{\eta}) = y_{\eta} |\vec{V}|_{\text{max}}
\]

Get the location of the maximum value of \( F(y_{\eta}) \), and compute \( F_{\text{wake}} \).

5b. Repeat step 5a using the location of \( |\vec{V}|_{\text{min}} \) as the origin for \( y_{\eta} \).

5c. Set the final value of \( F_{\text{wake}} \) equal to the one from step 5a or 5b that corresponds to the smaller value of \( y_{\text{max}} \). Set \( L_{\text{wake}} \) accordingly.
6. If a solid wall exists at $\eta = 0$ or at $\eta = 1$, but not both, or if neither $\eta$ boundary is a solid wall, compute $(\mu)_{\text{outer}}$ directly.

7. If both $\eta$ boundaries are solid walls, compute $(\mu)_{\text{outer}}$ by combining the two computed values of $F_{\text{wall}}$ using the averaging formula presented as equation (9.12) of Volume 1.

**Remarks**

1. The computation of $y_\text{n}$, the normal distance to the wall or wake centerline, is approximate. It is actually the straight-line distance between the interior grid point and the wall or wake centerline grid point.

2. The Cray search routines ISAMAX and ISAMIN are used in computing $|\bar{V}_{\text{max}}|$, $|\bar{V}_{\text{min}}|$, and $F_{\text{max}}$. The Cray search routine ISRCHEQ is used in determining the grid locations for debug printout.

3. If the maximum and minimum total velocities are equal, indicating a uniform flow along this particular $\xi$ line, their locations are arbitrarily set equal to the middle $\eta$ index, and the turbulent viscosity coefficient $(\mu)_{\text{outer}}$ is set equal to 0. This normally would occur only during the first time step in a case with uniform initial velocity profiles.

4. To avoid the possibility of floating point errors, the values of $|\bar{V}_{\text{max}}|$, $|\bar{V}_{\text{min}}|$, $F_{\text{max}}$, and $y_{\text{max}}$ are set to a minimum of $10^{-10}$.

5. This subroutine generates output for the IDEBUG(8) option.
Subroutine BLOUT2

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>TURBBL</td>
<td>ISAMAX ISAMIN ISRCHEQ</td>
<td>Compute outer layer turbulent viscosity, using the algebraic Baldwin-Lomax model, along constant ( \eta ) lines.</td>
</tr>
</tbody>
</table>

**Input**

- **APLUS**: Van Driest damping constant \( A^+ \).
- **CB**: Constant \( B \) in the Klebanoff intermittency factor.
- **CCLAU**: Clauser constant \( K \) in the Baldwin-Lomax outer region model.
- **CCP**: Constant \( C_{\mu} \) in the Baldwin-Lomax outer region model.
- **CKLEB, CKMIN**: Constants \( C_{Kleb} \) and \( (C_{Kleb})_{mn} \) in the Klebanoff intermittency factor.
- **CNA**: Exponent \( n \) in the formula used to average the two outer region \( \mu \) profiles that result when both boundaries in a coordinate direction are solid surfaces.
- **CWK**: Constant \( C_{\alpha} \) in the Baldwin-Lomax outer region model.
- **IDEBUG**: Debug flags.
- **IPRT1A, IPRT2A**: Indices for printout in the \( \xi \) and \( \eta \) directions.
- **IT**: Current time step number \( n \).
- **IWALL1**: Flags indicating whether or not the \( \xi \) boundaries are walls.
- **I2**: Grid index \( j \) in the \( \eta \) direction.
- **LWALL1**: Flags specifying wall locations for \( \xi \) boundaries.
- **MU**: Laminar coefficient of viscosity \( \mu_{r} \).
- **NOUT**: Unit number for standard output.
- **NPRT1, NPRT2**: Total number of indices for printout in the \( \xi \) and \( \eta \) directions.
- **N1**: Number of grid points \( N_1 \) in the \( \xi \) direction.
- **RER**: Reference Reynolds number \( Re \).
- **RHO, U, V, W**: Static density \( \rho \), and velocities \( u, v, \) and \( w \).
- **VORT**: Total vorticity magnitude.
- **X, Y**: Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \).

**Output**

- **LWAKE1**: Grid index \( i \) in the \( \xi \) direction used as the origin for computing length scales for free turbulent flows.
- **DUMMY**: Outer layer turbulent viscosity coefficient \( (\mu_{r})_{outer} \) along constant \( \eta \) lines.

**Description**

Subroutine BLOUT2 computes the outer layer turbulent viscosity coefficient \( (\mu_{r})_{outer} \) at a specified \( \eta \) location (i.e., due to walls at \( \xi = 0 \) and/or \( \xi = 1 \), or due to a free turbulent flow in the \( \eta \) direction) using the Proteus 2-D Programmer's Reference 4.0 Proteus Subprograms: BLOUT2 135
algebraic eddy viscosity model of Baldwin and Lomax (1978). The procedure is exactly analogous to that used in subroutine BLOUT1.
Subroutine BVUP (A,B,C,S,METX,METY,METT,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>BCGEN</td>
<td>Update first sweep boundary values after second sweep.</td>
</tr>
<tr>
<td></td>
<td>EQSTAT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SGEFA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SGESL</td>
<td></td>
</tr>
</tbody>
</table>

**Input**
- A, B, C: Coefficient submatrices A, B, and C.
- DXI: Computational grid spacing \( \Delta \xi \).
- IBVUP: Flags for updating boundary values from first sweep after second sweep; 0 if updating is not necessary, 1 if it is.
- * IHSTAG: Flag for constant stagnation enthalpy option.
- * ISWIRL: Flag for swirl in axisymmetric flow.
- JI: Inverse Jacobian of the nonorthogonal grid transformation; \( J^{-1} \).
- KBCPER: Flags for spatially periodic boundary conditions in the \( \xi \) and \( \eta \) directions; 0 for non-periodic, 1 for periodic.
- NEQ: Number of coupled equations being solved, \( N_Q \).
- NEQP: Dimensioning parameter specifying maximum number of coupled equations allowed.
- NPT1, NPT2: \( N_1 \) and \( N_2 \) for non-periodic boundary conditions, \( N_1 + 1 \) and \( N_2 + 1 \) for spatially periodic boundary conditions in \( \xi \) and \( \eta \).
- NR, NRU, NRV, NRW, NET: Array indices associated with the dependent variables \( \rho, \rho u, \rho v, \rho w, \) and \( E_T \).
- NVD, NPTSD: Leading two dimensions for the arrays A, P, C, and S.
- * N1, N2: Number of grid points \( N_1 \) and \( N_2 \) in the \( \xi \) and \( \eta \) directions.
- N1P: Parameter specifying the dimension size in the \( \xi \) direction.
- RHO, U, V, W, ET: Static density \( \rho \), velocities \( u, v, \) and \( w, \) and total energy \( E_T \) at time level \( n \) at all grid points.
- RHOL, UL, VL, WL, ETL: Static density \( \rho \), velocities \( u, v, \) and \( w, \) and total energy \( E_T \) at time level \( n + 1 \) at all interior grid points.
- S: Source term subvector S.
- XIX, XIY, XIT: Metric coefficients \( \xi_x, \xi_y \) (or \( \xi_r \) if axisymmetric), and \( \xi_t \).

**Output**
- DEL: Computational grid spacing for the sweep direction being updated.
- IBASE, ISTEP: Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- ISWEEP: ADI sweep number for sweep direction being updated.
- IV: Index in the "vectorized" direction, \( \iota \).
- METX, METY, METT: Derivatives of computational coordinate, for the sweep direction being updated, with respect to \( x, y \) (or \( r \) if axisymmetric), and \( t \).
NPTS
NV
RHOL, UL, VL, WL, ETL

Number of grid points \( N \) in the sweep direction being updated.
Number of grid points in the "vectorized" direction, \( N_v \).
Static density \( \rho \), velocities \( u, v, \) and \( w \), and total energy \( E_T \) at time level \( n + 1 \) at boundary points from first sweep.

Description

Subroutine BVUP updates boundary values from the first, or \( \xi \), sweep after the second, or \( \eta \), sweep. In general, this is necessary when gradient or extrapolation boundary conditions are used in the \( \xi \) direction. Some updating is also necessary when spatially periodic boundary conditions are used. The procedure is described in Section 7.3 of Volume 1 for all cases.

Remarks

1. The corner values of \( \rho \) and \( E_T \) are updated by linearly extrapolating from the two adjacent points in the \( \xi \) and \( \eta \) directions, and averaging the two results. Note that this extrapolation is done in computational space. Grid packing in either direction is thus not taken into account. The corner values of the velocities are updated by doing the same type of extrapolation. Instead of averaging, however, the extrapolated velocity whose absolute value is lower is used. This was done to maintain no-slip at duct inlets and exits.

2. Subroutines SGEFA and SGESL are Cray LINPACK routines. In general terms, if the Fortran arrays \( A \) and \( S \) represent \( A \) and \( S \), where \( A \) is a square \( N \) by \( N \) matrix and \( S \) is a vector with \( N \) elements, and if the leading dimension of the Fortran array \( A \) is \( LDA \), then the Fortran sequence

```fortran
    call sgefa (a,lda,n,ipvt,info)
    call sgels (a,lda,n,ipvt,s,0)
```

computes \( A^{-1}S \), storing the result in \( S \).
Subroutine COEFC (A,B,C,S,METX,METY,METT,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td></td>
<td>Compute coefficients and source term for the continuity equation.</td>
</tr>
</tbody>
</table>

**Input**

- **DEL**
  - Computational grid spacing in sweep direction.
- **DTAU**
  - Time step $\Delta t$.
- **DXI, DETA**
  - Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **ETAX, ETAY, ETAT**
  - Metric coefficients $\eta_x$, $\eta_y$ (or $\eta_z$ if axisymmetric), and $\eta_t$.
- **IAXI**
  - Flag for axisymmetric flow.
- **IBASE, STEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IHSTAG**
  - Flag for constant stagnation enthalpy option.
- **ISWEEP**
  - Current ADI sweep number.
- **ISWIRL**
  - Flag for swirl in axisymmetric flow.
- **IV**
  - Index in the "vectorized" direction, $i$.
- **I1, I2**
  - Grid indices $i$ and $j$, in the $\xi$ and $\eta$ directions.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$ (times the radius $r$ for axisymmetric flow.).
- **METX, METY, METT**
  - Derivatives of sweep direction computational coordinate with respect to $x$, $y$ (or $r$ if axisymmetric), and $t$.
- **NC**
  - Array index associated with the continuity equation.
- **NEQ**
  - Number of coupled equations being solved, $N_{eq}$.
- **NPTS**
  - Number of grid points in the sweep direction, $N$.
- **NR, NRU, NRV, NRW, NET**
  - Array indices associated with the dependent variables $\rho$, $\rho u$, $\rho v$, $\rho w$, and $E_r$.
- **NVD, NPTSD**
  - Leading two dimensions for the arrays A, B, C, S, METX, METY, and METT.
- **RAX**
  - 1 for two-dimensional planar flow, and the local radius $r$ for axisymmetric flow.
- **RHO, U, V**
  - Static density $\rho$, and velocities $u$ and $v$, at time level $n$.
- **RHOL**
  - Static density $\rho$ from previous ADI sweep.
- **THC**
  - Parameters $\theta_1$ and $\theta_2$ determining type of time differencing for the continuity equation.
- **XIX, XIY, XIT**
  - Metric coefficients $\xi_x$, $\xi_y$ (or $\xi_z$ if axisymmetric), and $\xi_t$.
- **Y**
  - Radial coordinate $r$ for axisymmetric flow.

**Output**

- **A, B, C**
  - Coefficient submatrices A, B, and C at interior points (row NC only).
Source term subvector S at interior points (element NC only).

**Description**

Subroutine COEFC computes the coefficients and source term for the continuity equation. Equations (7.5a-b) in Volume 1 represent, in vector form, the four governing difference equations for the two ADI sweeps for 2-D planar flow. The elements of the inviscid flux vectors \( \hat{\mathbf{E}} \) and \( \hat{\mathbf{F}} \) are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors \( \hat{\mathbf{E}}_{visc}, \hat{\mathbf{E}}_{visc2}, \) etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices \( \partial E_i / \partial \hat{Q}, \partial \hat{E}_i / \partial \hat{Q} \), etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the continuity equation for 2-D planar flow may be written for the two ADI sweeps as\(^{17}\)

**Sweep 1 (\( \xi \) direction)**

\[
\Delta \hat{\rho}_i^* + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2\Delta \xi} \begin{bmatrix}
(\frac{\partial \hat{E}_1}{\partial \hat{Q}})_i^n \Delta \hat{Q}_{i+1}^* - (\frac{\partial \hat{E}_1}{\partial \hat{Q}})_i^{n-1} \\
\end{bmatrix} = 0
\]

**Sweep 2 (\( \eta \) direction)**

\[
\Delta \hat{\rho}_j^* + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2\Delta \eta} \begin{bmatrix}
(\frac{\partial \hat{F}_1}{\partial \hat{Q}})_j^n \Delta \hat{Q}_{j+1}^* - (\frac{\partial \hat{F}_1}{\partial \hat{Q}})_j^{n-1} \\
\end{bmatrix} = \Delta \hat{\rho}^*
\]

In the above equations, the subscripts \( i \) and \( j \) represent grid point indices in the \( \xi \) and \( \eta \) directions. For notational convenience, terms without an explicitly written \( i \) or \( j \) subscript are understood to be at \( i \) or \( j \).

The vector of dependent variables is

\[
\hat{\mathbf{Q}} = \frac{1}{J} [\rho \quad \rho u \quad \rho v \quad E_T]^T
\]

The appropriate elements of the flux vectors are given by

\[
\hat{\mathbf{E}}_1 = \frac{1}{J} [\rho u \xi_x + \rho v \xi_y + \rho \xi_z]
\]

\[
\hat{\mathbf{F}}_1 = \frac{1}{J} [\rho u \eta_x + \rho v \eta_y + \rho \eta_z]
\]

The elements of the Jacobian coefficient matrix \( \partial \hat{\mathbf{E}} / \partial \hat{\mathbf{Q}} \) for the continuity equation are

\[
\frac{\partial \hat{\mathbf{E}}_1}{\partial \hat{\mathbf{Q}}} = [\xi_t \quad \xi_x \quad \xi_y \quad 0]
\]

\(^{17}\) These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy \( E_T \) would not appear as a dependent variable, and the Jacobian coefficient matrices would have only three elements.
The Jacobian coefficient matrix \( \partial F_i / \partial \hat{Q} \) has the same form as \( \partial E_i / \partial \hat{Q} \), but with \( \xi \) replaced by \( \eta \).

As an example of how these equations are translated into Fortran, consider the \( \Delta (\rho u / J) \) term on the left hand side for the first sweep. This is the second element of \( \hat{Q} \), so using the second element in \( \partial E_i / \partial \hat{Q} \) we get

\[
A(IV, I, NC, NRU) = -\frac{\theta_1 (\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left( \hat{E}_{i-1,j} \right) \\
B(IV, I, NC, NRU) = 0 \\
C(IV, I, NC, NRU) = \frac{\theta_1 (\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left( \hat{E}_{i+1,j} \right)
\]

The equations for axisymmetric flow are developed in Appendix B of Volume 1. The axisymmetric continuity equation for the two ADI sweeps is given by\textsuperscript{18}

**Sweep 1 (\( \xi \) direction)**

\[
\Delta \rho_i^* + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2\Delta \xi} \frac{1}{r} \left[ \left( r \frac{\partial \hat{E}_1}{\partial \hat{Q}} \right)^n \Delta \hat{Q}_{i+1}^* - \left( r \frac{\partial \hat{E}_1}{\partial \hat{Q}} \right)^n \Delta \hat{Q}_{i-1}^* \right] = \\
-\frac{\Delta \tau}{1 + \theta_2} \frac{1}{r} \left[ \delta_{\xi} (r \hat{E}_i) + \delta_{\eta} (r \hat{F}_i) \right] + \frac{\theta_2}{1 + \theta_2} \Delta \rho^{n-1}
\]

**Sweep 2 (\( \eta \) direction)**

\[
\Delta \rho_j^n + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2\Delta \eta} \frac{1}{r} \left[ \left( r \frac{\partial \hat{F}_1}{\partial \hat{Q}} \right)^n \Delta \hat{Q}_{j+1}^n - \left( r \frac{\partial \hat{F}_1}{\partial \hat{Q}} \right)^n \Delta \hat{Q}_{j-1}^n \right] = \Delta \rho^*
\]

where now

\[
\hat{Q} = \frac{1}{J} [\rho \quad \rho u \quad \rho v \quad \rho w \quad E_T]^T \\
\hat{E}_1 = \frac{1}{J} [\rho \mu \xi_x + \rho \nu \xi_y + \rho \xi_z] \\
\hat{F}_1 = \frac{1}{J} [\rho \mu \eta_x + \rho \nu \eta_y + \rho \eta_z] \\
\frac{\partial \hat{E}_1}{\partial \hat{Q}} = [\xi_x \quad \xi_y \quad \xi_z \quad 0 \quad 0]
\]

As in 2-D planar flow, the Jacobian coefficient matrix \( \partial \hat{F}_i / \partial \hat{Q} \) has the same form as \( \partial \hat{E}_i / \partial \hat{Q} \), but with \( \xi \) replaced by \( \eta \).

\textsuperscript{18} These equations are written for the general case with swirl. For a non-swirl case, the swirl momentum \( \rho w \) would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.
Note that the equations for 2-D planar and axisymmetric flow are very similar. In the axisymmetric equations, the radius \( r \) appears as an additional coefficient in front of the flux vectors \( \mathbf{E} \) and \( \mathbf{F} \), and in front of the Jacobian coefficient matrices \( \frac{\partial \mathbf{E}}{\partial Q} \) and \( \frac{\partial \mathbf{F}}{\partial Q} \). In addition, \( 1/r \) appears in front of every term in the equation except the \( \Delta \rho \) terms. In Proteus, the Fortran variables are defined in such a way that, for many terms, the same coding can be used for both 2-D planar and axisymmetric flow. Unfortunately, this may make some of the coding a little confusing. It is hoped that this detailed description, when compared with the source listing, will help make things clear.

In COEFC, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second ADI sweep have exactly the same form as for the first sweep, but with \( \zeta \) replaced by \( \eta \). By defining \( \text{DEL} \), \( \text{METX} \), \( \text{METY} \), and \( \text{METT} \) as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for both sweeps. The variable \( \text{RAX} \) is equal to 1 for 2-D planar flow, and the radius \( r \) for axisymmetric flow. This adds the \( r \) in front of the Jacobian coefficient matrices for axisymmetric flow, but has no effect for 2-D planar flow. The \( 1/r \) coefficient in front of each term will be added later. In this section of code, the coefficient of \( \Delta \rho \) (part of \( B(IV,I,NC,NR) \)) is set equal to \( r \), not 1 as it should be. This will be corrected later.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms. These formulas are presented in Section 5.0 of Volume 1. For axisymmetric flow, the Fortran variable \( JI \), which is normally defined as \( 1/J \), is temporarily redefined as \( r/J \) before the COEF routines are called. This automatically accounts for the \( r \) coefficient in front of all the flux vectors in the source term. The \( 1/r \) coefficient in front of each term will be added later. This definition of \( JI \) adds an \( r \) in front of the \( \Delta \rho \) term that should not be there. This will also be corrected later.

The coding for the source term for the second sweep, which consists only of \( \Delta \rho \), comes next. The definition of \( JI \) also adds an \( r \) in front of this term that should not be there.

And finally, for axisymmetric flow, the entire equation is divided by the local radius \( r \). This adds the \( 1/r \) coefficient where it should be added, and removes the \( r \) in front of the \( \Delta \rho \) terms.

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables \( A, B, C, \) and \( S \) may be confusing. The first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. For sections of the code that apply to both sweeps (i.e., the implicit terms and the division by \( r \) at the end), the first two subscripts are written as \( (IV,I) \). For sections of the code that apply only to the first sweep, the first two subscripts are written as \( (I2,I1) \). For sections that apply only to the second sweep, they are written as \( (I1,I2) \). The third subscript on \( A, B, C, \) and \( S \) corresponds to the equation. And, for \( A, B, \) and \( C \), the fourth subscript corresponds to the dependent variable for which \( A, B, \) or \( C \) is a coefficient.
Subroutine COEFE (A,B,C,S,METX,METY,METT,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td></td>
<td>Compute coefficients and source term for the energy equation.</td>
</tr>
</tbody>
</table>

### Input

**DEL**  
Computational grid spacing in sweep direction.

**DPDRHO, DPDRU, DPDRV, DPDRW, DPDET**  
Derivatives $\partial p/\partial \rho$, $\partial p/\partial (\rho u)$, $\partial p/\partial (\rho v)$, $\partial p/\partial (\rho w)$, and $\partial p/\partial \varepsilon_T$.

**DTAU**  
Time step $\Delta \tau$.

**DTDHRHO, DTDRU, DTDVR, DTDRW, DTDDET**  
Derivatives $\partial T/\partial \rho$, $\partial T/\partial (\rho u)$, $\partial T/\partial (\rho v)$, $\partial T/\partial (\rho w)$, and $\partial T/\partial \varepsilon_T$.

**DXI, DETA**  
Computational grid spacing $\Delta \xi$ and $\Delta \eta$.

**ETAX, ETAY, ETAT**  
Metric coefficients $\eta_x$, $\eta_y$ (or $\eta_z$ if axisymmetric), and $\eta_r$.

**IAXI**  
Flag for axisymmetric flow.

**IBASE, ISTEP**  
Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.

**IEULER**  
Flag for Euler calculation.

**ISWEEP**  
Current ADI sweep number.

**ISWIRL**  
Flag for swirl in axisymmetric flow.

**ITHIN**  
Flags for thin-layer option.

**IV**  
Index in the "vectorized" direction, $i$.

**I1, I2**  
Grid indices $i$ and $j$ in the $\xi$ and $\eta$ directions.

**JI**  
Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$ (times the radius $r$ for axisymmetric flow.)

**METX, METY, METT**  
Derivatives of sweep direction computational coordinate with respect to $x$, $y$ (or $r$ if axisymmetric), and $t$.

**MU, LA, KT**  
Effective coefficient of viscosity $\mu$, effective second coefficient of viscosity $\lambda$, and effective coefficient of thermal conductivity $k$ at time level $n$.

**NEN**  
Array index associated with the energy equation.

**NEQ**  
Number of coupled equations being solved, $N_{eq}$.

**NPTS**  
Number of grid points in the sweep direction, $N$.

**NR, NRU, NRV, NRW, NET**  
Array indices associated with the dependent variables $\rho$, $\rho u$, $\rho v$, $\rho w$, and $\varepsilon_T$.

**NVD, NPTSD**  
Leading two dimensions for the arrays A, B, C, S, METX, METY, and METT.

**P, T**  
Static pressure $p$ and temperature $T$ at time level $n$.

**P RR**  
Reference Prandtl number $Pr_z$.

**RAX**  
1 for two-dimensional planar flow, and the local radius $r$ for axisymmetric flow.

**RER**  
Reference Reynolds number $Re_z$.
RHO, U, V, W, ET

Static density \( \rho \), velocities \( u, v, \) and \( w \), and total energy \( E_T \) at time level \( n \).

THE

Parameters \( \theta_1, \theta_2, \) and \( \theta_3 \) determining type of time differencing for the energy equation.

TL

Static temperature \( T \) from previous ADI sweep.

UL, VL, WL, ETL

Velocities \( u, v, \) and \( w \), and total energy \( E_T \) from previous ADI sweep.

XIX, XHY, XIT

Metric coefficients \( \xi, \xi \) (or \( \xi \), if axisymmetric), and \( \xi \).

Y

Radial coordinate \( r \) for axisymmetric flow.

Output

A, B, C

Coefficient submatrices \( A, B, \) and \( C \) at interior points (row NEN only).

S

Source term subvector \( S \) at interior points (element NEN only).

Description

Subroutine COEFE computes the coefficients and source term for the energy equation. Equations (7.5a-b) in Volume 1 represent, in vector form, the four governing difference equations for the two ADI sweeps for 2-D planar flow. The elements of the inviscid flux vectors \( \hat{E} \) and \( \hat{F} \) are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors \( \hat{E}_v, \hat{E}_{v2}, \) etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices \( \frac{\partial \hat{E}}{\partial \hat{Q}}, \frac{\partial \hat{E}_v}{\partial \hat{Q}}, \) etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the energy equation for 2-D planar flow may be written for the two ADI sweeps as

Sweep 1 (\( \xi \) direction)

\[
\Delta (\hat{E}_7^*)_i + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2 \Delta \xi^2} \left[ \left( \frac{\partial \hat{E}_4}{\partial \hat{Q}} \right)^n \Delta \hat{Q}^*_i - \left( \frac{\partial \hat{E}_4}{\partial \hat{Q}} \right)^{i-1} \Delta \hat{Q}^*_i \right] 
\]

\[
- \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2 \Delta \xi^2} \left[ (f_i + f_i) g_i^n - (f_i + f_i + 1) g_i^n \Delta \hat{Q}^*_i + (f_i + f_i + 1) g_i^n \Delta \hat{Q}^*_i + (f_i + f_i + 1) g_i^n \Delta \hat{Q}^*_i \right] = 
\]

\[
- \frac{\Delta \tau}{1 + \theta_2} \left( \delta_\xi \hat{E}_4 + \delta_\eta \hat{F}_4 \right)^n + \frac{\Delta \tau}{1 + \theta_2} \left[ \delta_\xi (\hat{E}_v)_4 + \delta_\eta (\hat{F}_v)_4 \right]^n 
\]

\[
+ \frac{(1 + \theta_3) \Delta \tau}{1 + \theta_2} \left[ \delta_\xi (\hat{E}_v)_4 + \delta_\eta (\hat{F}_v)_4 \right]^n - \frac{\theta_3 \Delta \tau}{1 + \theta_2} \left[ \delta_\xi (\hat{E}_v)_4 + \delta_\eta (\hat{F}_v)_4 \right]^n - 1 + \frac{\theta_2}{1 + \theta_2} \Delta \hat{E}_T^n
\]
Sweep 2 ($\eta$ direction)

$$
\Delta(\hat{E}_T)_{j+1} + \frac{\theta_1 \Delta \tau}{(1 + \theta_2) 2 \Delta \eta} \left[ \left( \frac{\partial \hat{F}_4}{\partial \hat{Q}} \right)_{j+1} \Delta \hat{Q}_{j+1}^\eta - \left( \frac{\partial \hat{F}_4}{\partial \hat{Q}} \right)_{j-1} \Delta \hat{Q}_{j-1}^\eta \right] = \\
- \frac{\theta_1 \Delta \tau}{(1 + \theta_2) 2 \Delta \eta} \left[ (f_j + f_j) \hat{g}_{j-1}^\eta \Delta \hat{Q}_{j-1}^\eta - (f_j + 2f_j + f_{j+1}) \hat{g}_{j}^\eta \Delta \hat{Q}_j^\eta + (f_j + f_{j+1}) \hat{g}_{j+1}^\eta \Delta \hat{Q}_{j+1}^\eta \right]
$$

In the above equations, the subscripts $i$ and $j$ represent grid point indices in the $\xi$ and $\eta$ directions. For notational convenience, terms without an explicitly written $i$ or $j$ subscript are understood to be at $i$ or $j$.

On the left hand side, $f$ is the coefficient of $\partial/\partial \xi$ (or $\partial/\partial \eta$, depending on the sweep) in the $\partial \hat{E}_\nu / \partial \hat{Q}$ (or $\partial \hat{F}_\nu / \partial \hat{Q}$) Jacobian coefficient matrix. Similarly, $g$ is the term in the parentheses following $\partial/\partial \xi$ (or $\partial/\partial \eta$) in the $\partial \hat{E}_\nu / \partial \hat{Q}$ (or $\partial \hat{F}_\nu / \partial \hat{Q}$) Jacobian coefficient matrix.

The vector of dependent variables is

$$\hat{Q} = \frac{1}{j} [\rho \quad \rho u \quad \rho v \quad E_T]^T$$

The appropriate elements of the inviscid flux vectors are given by

$$\hat{E}_4 = \frac{1}{J} \left[ (E_T + p) u^2_x + (E_T + p) v^2_y + E_T \xi_T \right]$$

$$\hat{F}_4 = \frac{1}{J} \left[ (E_T + p) u^2_x + (E_T + p) v^2_y + E_T \eta_T \right]$$

The appropriate elements of the non-cross derivative viscous flux vectors are

$$\begin{align*}
(\hat{E}_\nu)_4 &= \frac{1}{J} \frac{1}{Re_r} \left\{ \frac{(2\mu + \lambda)}{2} \left[ \hat{\xi}_x^2 (u^2)_x + \hat{\xi}_y^2 (v^2)_y \right] + \left( \mu + \lambda \right) \hat{\xi}_x \hat{\xi}_y \left( \nu \nu \right)_x + \\
&\quad + \frac{\mu}{2} \left[ \hat{\xi}_x^2 (u^2)_x + \hat{\xi}_y^2 (v^2)_y \right] + \frac{k_{Pr_r}}{(\hat{\xi}_x^2 + \hat{\xi}_y^2) T_T} \right\} \\
(\hat{F}_\nu)_4 &= \frac{1}{J} \frac{1}{Re_r} \left\{ \frac{(2\mu + \lambda)}{2} \left[ \hat{\eta}_x^2 (u^2)_x + \hat{\eta}_y^2 (v^2)_y \right] + \left( \mu + \lambda \right) \hat{\eta}_x \hat{\eta}_y \left( \nu \nu \right)_\eta + \\
&\quad + \frac{\mu}{2} \left[ \hat{\eta}_x^2 (u^2)_x + \hat{\eta}_y^2 (v^2)_y \right] + \frac{k_{Pr_r}}{(\hat{\eta}_x^2 + \hat{\eta}_y^2) T_T} \right\}
\end{align*}$$

And the appropriate elements of the cross derivative viscous flux vectors are

$$\begin{align*}
(\hat{E}_\nu)_4 &= \frac{1}{J} \frac{1}{Re_r} \left[ 2 \mu (\xi_x u \eta_x + \xi_y v \eta_y) + \lambda \xi_x (\eta_x u \eta_x + \eta_x v \eta_y) + \lambda \xi_y (\xi_x u \eta_y + \eta_y v \nu_y) \\
&\quad + \mu \xi_x (\eta_x u \eta_x + \eta_x v \eta_y) + \mu \xi_y (\xi_x u \eta_y + \eta_y v \nu_y) + \frac{k}{Pr_r} \left( \xi_x \eta_x + \xi_y \eta_y \right) T_T \right]
\end{align*}$$
\[
(\hat{\mathbf{F}}_\nu)_4 = \frac{1}{f} \frac{1}{Re} \left[ 2\mu (\eta_x \xi_x u_{\xi} + \eta_y \xi_y v_{\xi}) + \lambda \eta_x (\xi_x u_{\xi} + \xi_y v_{\xi}) + \lambda \eta_y (\xi_y u_{\xi} + \xi_y v_{\xi}) \\
+ \mu \eta_x (\xi_y v_{\xi} + \xi_x u_{\xi}) + \mu \eta_y (\xi_y v_{\xi} + \xi_x u_{\xi}) + \frac{k}{Pr} (\eta_x \xi_x + \eta_y \xi_y) T_{\xi} \right]
\]

The elements of the Jacobian coefficient matrix \( \partial \hat{\mathbf{E}}_4 / \partial \hat{\mathbf{Q}} \) for the inviscid terms in the energy equation are

\[
\frac{\partial \hat{\mathbf{E}}_4}{\partial \hat{\mathbf{Q}}} = \left[ -f_1 \left( f_2 - \frac{\partial p}{\partial \rho} \right) f_2 \xi_x + f_1 \frac{\partial p}{\partial (\rho u)} f_2 \xi_y + f_1 \frac{\partial p}{\partial (\rho v)} \xi_t + f_1 \left( 1 + \frac{\partial p}{\partial E_T} \right) \right]
\]

where \( f_1 = u_{\xi_x} + v_{\xi_y} \) and \( f_2 = (E_T + \rho)/\rho \).

The elements of the Jacobian coefficient matrix \( \partial \hat{\mathbf{E}}_{\nu_1} / \partial \hat{\mathbf{Q}} \) for the viscous terms are

\[
\frac{\partial \hat{\mathbf{E}}_{\nu_1}}{\partial \hat{\mathbf{Q}}} = \frac{1}{Re} \left[ \left( \frac{\partial \hat{\mathbf{E}}_{\nu_1}}{\partial \hat{\mathbf{Q}}} \right)_1 \left( \frac{\partial \hat{\mathbf{E}}_{\nu_1}}{\partial \hat{\mathbf{Q}}} \right)_2 \left( \frac{\partial \hat{\mathbf{E}}_{\nu_1}}{\partial \hat{\mathbf{Q}}} \right)_3 \alpha_0 \frac{\partial}{\partial \xi} \left( \frac{\partial \hat{T}}{\partial E_T} \right) \right]
\]

where

\[
\left( \frac{\partial \hat{\mathbf{E}}_{\nu_1}}{\partial \hat{\mathbf{Q}}} \right)_1 = - \alpha_{xx} \frac{\partial}{\partial \xi} \left( \frac{u^2}{\rho} \right) - \alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{v^2}{\rho} \right) - 2\alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{uv}{\rho} \right) + \alpha_0 \frac{\partial}{\partial \xi} \left( \frac{\partial T}{\partial \rho} \right)
\]

\[
\left( \frac{\partial \hat{\mathbf{E}}_{\nu_1}}{\partial \hat{\mathbf{Q}}} \right)_2 = \alpha_{xx} \frac{\partial}{\partial \xi} \left( \frac{u^2}{\rho} \right) + \alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{v^2}{\rho} \right) + \alpha_0 \frac{\partial}{\partial \xi} \left( \frac{\partial T}{\partial (\rho u)} \right)
\]

\[
\left( \frac{\partial \hat{\mathbf{E}}_{\nu_1}}{\partial \hat{\mathbf{Q}}} \right)_3 = \alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{u^2}{\rho} \right) + \alpha_{yy} \frac{\partial}{\partial \xi} \left( \frac{v^2}{\rho} \right) + \alpha_0 \frac{\partial}{\partial \xi} \left( \frac{\partial T}{\partial (\rho v)} \right)
\]

\[
\alpha_{xx} = (2\mu + \lambda) \xi_x^2 + \mu \xi_y^2
\]

\[
\alpha_{xy} = \mu \xi_x^2 + (2\mu + \lambda) \xi_y^2
\]

\[
\alpha_{yy} = (\mu + \lambda) \xi_x \xi_y
\]

\[
\alpha_0 = \frac{k}{Pr} (\xi_x^2 + \xi_y^2)
\]

The Jacobian coefficient matrices \( \partial \hat{\mathbf{F}}_4 / \partial \hat{\mathbf{Q}} \) and \( \partial (\hat{\mathbf{F}}_{\nu_1})_4 / \partial \hat{\mathbf{Q}} \) have the same form as \( \partial \hat{\mathbf{E}}_4 / \partial \hat{\mathbf{Q}} \) and \( \partial \hat{\mathbf{E}}_{\nu_1} / \partial \hat{\mathbf{Q}} \), but with \( \xi \) replaced by \( \eta \).

As an example of how these equations are translated into Fortran, consider the \( \Delta(\rho u/L) \) term on the left hand side for the first sweep. This is the second element of \( \hat{\mathbf{Q}} \), so using the second element in \( \partial \hat{\mathbf{E}}_4 / \partial \hat{\mathbf{Q}} \) we get for the inviscid term
\[ A(IV,I,NEN,NRU) = -\frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left\{ \left( \frac{E_T + p}{\rho} - \xi_x \right)_{i-1,j} + \left[ (u_x + v_y) \frac{\partial p}{\partial (\rho u)} \right]_{i-1,j} \right\} \]

\[ B(IV,I,NEN,NRU) = 0 \]

\[ C(IV,I,NEN,NRU) = \frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left\{ \left( \frac{E_T + p}{\rho} - \xi_x \right)_{i+1,j} + \left[ (u_x + v_y) \frac{\partial p}{\partial (\rho u)} \right]_{i+1,j} \right\} \]

For the viscous terms on the left hand side, we use the second element in \( \partial (\hat{E}_n)_{i,j}/\partial \hat{Q} \), which is

\[
\frac{1}{Re} \left[ a_{xx} \frac{\partial}{\partial \xi_x} \left( \frac{u}{\rho} \right) + a_{xy} \frac{\partial}{\partial \xi} \left( \frac{v}{\rho} \right) + a_{0} \frac{\partial}{\partial \xi} \left( \frac{\partial T}{\partial (\rho u)} \right) \right]
\]

There are three terms in that element. Thus, in turn, \( f = a_{xx}/Re \), \( a_{xy}/Re \), and \( a_{0}/Re \), and \( g = u/\rho, v/\rho \), and \( \partial T/\partial (\rho u) \). To add the viscous contribution to this part of the A coefficient submatrix, we therefore set

\[
A(IV,I,NEN,NRU) = A(IV,I,NEN,NRU) - \left( \frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2(\Delta \xi)^2Re} \right) \left\{ \left( \frac{\partial E_n}{\partial Q} \right)_{i,j} \right\}
\]

Similar equations may be written for the B and C coefficient submatrices.

The equations for axisymmetric flow are developed in Appendix B of Volume 1. The axisymmetric energy equation for the two ADI sweeps is given by\(^{19}\)

**Sweep 1 (z direction)**

\[
\Delta(\hat{E}_T)_{i}+ = \frac{\theta_1\Delta \tau}{(1 + \theta_2)2\Delta \xi} \left[ \frac{\partial E_T}{\partial Q} \right]_{i}^{n} - \left[ \frac{\partial E_T}{\partial Q} \right]_{i-1}^{n} - \frac{\theta_1\Delta \tau}{(1 + \theta_2)2(\Delta \xi)^2} \left[ \left( \xi_{i-1,i} - \xi_{i,i} \right) - \left( \xi_{i,i} - \xi_{i+1,i} \right) \right]
\]

**Sweep 2 (η direction)**

\[
\Delta(\hat{E}_T)_{j}+ = \frac{\theta_1\Delta \tau}{(1 + \theta_2)2\Delta \eta} \left[ \frac{\partial E_T}{\partial Q} \right]_{j}^{n} - \left[ \frac{\partial E_T}{\partial Q} \right]_{j-1}^{n} - \frac{\theta_1\Delta \tau}{(1 + \theta_2)2(\Delta \eta)^2} \left[ \left( \xi_{j-1,j} - \xi_{j,j} \right) - \left( \xi_{j,j} - \xi_{j+1,j} \right) \right]
\]

---

\(^{19}\) These equations are written for the general case with swirl. For a non-swirl case, the swirl momentum \( \rho w \) would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.
where now

$$\dot{Q} = \frac{1}{J} \begin{bmatrix} \rho u & \rho v & \rho w & E_T \end{bmatrix}^T$$

$$\dot{E}_S = \frac{1}{J} \begin{bmatrix} (E_T + p)u \xi_x + (E_T + p)v \xi_r + E_T \xi_z \end{bmatrix}$$

$$\dot{F}_S = \frac{1}{J} \begin{bmatrix} (E_T + p)u \eta_x + (E_T + p)v \eta_r + E_T \eta_z \end{bmatrix}$$

$$(\dot{E}_V)_S = \frac{1}{J Re} \left\{ \frac{(2\mu + \lambda)}{2} \left[ \xi_x^2 (u^2)_x + \xi_r^2 (v^2)_r \right] + (\mu + \lambda) \xi_x \xi_r (uv)_x + \lambda \xi_r \frac{r_x}{r} (\xi_x v^2 + \xi_r u v) \\
+ \frac{\mu}{2} \left[ \xi_x^2 (v^2 + w^2)_x + \xi_r^2 (u^2 + w^2)_r \right] + \frac{k}{Pr} \left( \xi_x^2 + \xi_r^2 \right) T \right\}$$

$$(\dot{F}_V)_S = \frac{1}{J Re} \left\{ \frac{(2\mu + \lambda)}{2} \left[ \eta_x^2 (u^2)_x + \eta_r^2 (v^2)_r \right] + (\mu + \lambda) \eta_x \eta_r (uv)_x + \lambda \eta_r \frac{r_x}{r} (\eta_x v^2 + \eta_r u v) \\
+ \frac{\mu}{2} \left[ \eta_x^2 (v^2 + w^2)_x + \eta_r^2 (u^2 + w^2)_r \right] + \frac{k}{Pr} \left( \eta_x^2 + \eta_r^2 \right) T \right\}$$

$$(\dot{E}_V)_S = \frac{1}{J Re} \left[ 2\mu (\xi_x^2 u u + \xi_r^2 v v) + \lambda \xi_x (\xi_x u u + \eta_r v v) + \lambda \xi_r (\xi_r u u + \eta_x v v) + \lambda \eta_r \frac{u}{v} (\xi_x u + \xi_r v) \eta \\
+ \mu (\xi_x^2 u u + \xi_r^2 v v) + \lambda \xi_x (\xi_x u u + \eta_r v v) + \lambda \xi_r (\xi_r u u + \eta_x v v) + \lambda \eta_r \frac{u}{v} (\xi_x u + \xi_r v) \eta \\
+ \frac{k}{Pr} (\xi_x^2 + \xi_r^2) T \right]$$

$$(\dot{F}_V)_S = \frac{1}{J Re} \left[ 2\mu (\eta_x^2 u u + \xi_r^2 v v) + \lambda \eta_x (\xi_x u u + \eta_r v v) + \lambda \eta_r (\xi_r u u + \eta_x v v) + \lambda \eta_r \frac{u}{v} (\xi_x u + \xi_r v) \eta \\
+ \mu (\eta_x^2 u u + \xi_r^2 v v) + \lambda \eta_x (\xi_x u u + \eta_r v v) + \lambda \eta_r (\xi_r u u + \eta_x v v) + \lambda \eta_r \frac{u}{v} (\xi_x u + \xi_r v) \eta \\
+ \frac{k}{Pr} (\eta_x^2 + \eta_r^2) T \right]$$

The elements of the Jacobian coefficient matrix $\partial \dot{E}/\partial \dot{Q}$ for the inviscid terms in the axisymmetric form of the energy equation are

$$\frac{\partial \dot{E}_S}{\partial Q} = \left[ \begin{array}{c} -f_1 f_2 \frac{\partial p}{\partial \rho} \ f_2 \xi_x + f_1 \frac{\partial p}{\partial (\rho u)} \ f_2 \xi_r + f_1 \frac{\partial p}{\partial (\rho v)} \ f_1 \frac{\partial p}{\partial (\rho w)} \end{array} \right]$$

where $f_1 = u \xi_x + v \xi_r$, and $f_2 = (E_T + p)/\rho$.

The elements of the Jacobian coefficient matrix $\partial \dot{E}_V/\partial \dot{Q}$ for the viscous terms are

$$\frac{\partial \dot{E}_V}{\partial Q} = \frac{1}{Re} \left[ \begin{array}{c} \left( \frac{\partial \dot{E}_V}{\partial Q} \right)_{11} \left( \frac{\partial \dot{E}_V}{\partial Q} \right)_{12} \left( \frac{\partial \dot{E}_V}{\partial Q} \right)_{13} \left( \frac{\partial \dot{E}_V}{\partial Q} \right)_{14} \end{array} \right]$$

where

148 4.0 Proteus Subprograms: COEFE Proteus 2-D Programmer’s Reference
\[
\left( \frac{\partial \hat{E}_{V_1}}{\partial \hat{Q}} \right) = -\alpha_{xx} \frac{\partial}{\partial \xi} \left( \frac{u^2}{\rho} \right) - \alpha_{rr} \frac{\partial}{\partial \xi} \left( \frac{v^2}{\rho} \right) - \alpha_{zz} \frac{\partial}{\partial \xi} \left( \frac{w^2}{\rho} \right)
- 2\alpha_{xr} \frac{\partial}{\partial \xi} \left( \frac{\nu w}{\rho} \right) - 2\alpha_{rr} \frac{\nu^2}{\rho} r_z - 2\alpha_{xr} \frac{\nu w}{\rho} r_z + a_0 \frac{\partial}{\partial \xi} \left( \frac{\partial T}{\partial \rho} \right)
\]

\[
\left( \frac{\partial \hat{E}_{V_1}}{\partial \hat{Q}} \right) = \alpha_{xx} \frac{\partial}{\partial \xi} \left( \frac{u}{\rho} \right) + \alpha_{xr} \frac{\partial}{\partial \xi} \left( \frac{v}{\rho} \right) + \alpha_{rr} \frac{\partial}{\partial \xi} \left( \frac{v}{\rho} r_z \right) + a_0 \frac{\partial}{\partial \xi} \left( \frac{\partial T}{\partial (\rho\nu)} \right)
\]

\[
\left( \frac{\partial \hat{E}_{V_1}}{\partial \hat{Q}} \right) = \alpha_{xx} \frac{\partial}{\partial \xi} \left( \frac{w}{\rho} \right) + \alpha_{rr} \frac{\partial}{\partial \xi} \left( \frac{w}{\rho} r_z \right) + a_0 \frac{\partial}{\partial \xi} \left( \frac{\partial T}{\partial (\rho w)} \right)
\]

\[
\alpha_{xx} = (2\mu + \lambda) \xi_x^2 + \mu \xi_r^2
\]
\[
\alpha_{rr} = \mu \xi_x^2 + (2\mu + \lambda) \xi_r^2
\]
\[
\alpha_{zz} = \mu \xi_x^2 + \mu \xi_r^2
\]
\[
\alpha_{xr} = (\mu + \lambda) \xi_x \xi_r
\]
\[
\alpha_{xr} = \frac{\lambda}{r} \xi_x \xi_r
\]
\[
\alpha_{rr} = \frac{\lambda}{r} \xi_r^2
\]
\[
a_0 = \frac{k}{Pr} (\xi_x^2 + \xi_r^2)
\]

As in 2-D planar flow, the Jacobian coefficient matrices \(\frac{\partial \hat{F}_s}{\partial \hat{Q}}\) and \(\frac{\partial (\hat{F}_{V_1})}{\partial \hat{Q}}\) have the same form as \(\frac{\partial \hat{E}_s}{\partial \hat{Q}}\) and \(\frac{\partial (\hat{E}_{V_1})}{\partial \hat{Q}}\), but with \(\xi\) replaced by \(\eta\).

Note that the equations for 2-D planar and axisymmetric flow are very similar. In the axisymmetric equations there are some additional terms involving the radius \(r\) in the viscous flux vectors, with corresponding terms in the Jacobian coefficient matrices. The radius \(r\) appears as an additional coefficient in front of the flux vectors \(\hat{E}, \hat{E}_{V_1}\), etc., and in front of the Jacobian coefficient matrices \(\frac{\partial \hat{E}_s}{\partial \hat{Q}}, \frac{\partial (\hat{E}_{V_1})}{\partial \hat{Q}}\), etc. In addition, \(1/r\) appears in front of every term in the equation except the \(\Delta \hat{E}_T\) terms. In Proteus, the Fortran variables are defined in such a way that, for many terms, the same coding can be used for both 2-D planar and axisymmetric flow. Unfortunately, this may make some of the coding a little confusing. It is hoped that this detailed description, when compared with the source listing, will help make things clear.

In COEFE, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second ADI sweep have exactly the same form as for the first sweep, but with \(\xi\) replaced by \(\eta\). By defining DEL, METX, METY, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for both sweeps. The variable RAX is equal to 1 for 2-D planar flow, and the radius \(r\) for axisymmetric flow. This adds the \(r\) in front of the Jacobian coefficient matrices for axisymmetric flow, but has no effect for 2-D planar flow. The \(1/r\) coefficient in front of each term will be
added later. In this section of code, the coefficient of $\Delta \hat{E}_r$ (part of $B(IV, I, NEN, NET)$) is set equal to $r$, not 1 as it should be. This will be corrected later.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms. These formulas are presented in Section 5.0 of Volume 1. For axisymmetric flow, the Fortran variable $JI$, which is normally defined as $1/J$, is temporarily redefined as $r/J$ before the COEF routines are called. This automatically accounts for the $r$ coefficient in front of all the flux vectors in the source term. The $1/r$ coefficient in front of each term will be added later. This definition of $JI$ adds an $r$ in front of the $\Delta \hat{E}_r^{-1}$ term that should not be there. This will also be corrected later.

The coding for the source term for the second sweep, which consists only of $\Delta \hat{E}_r$, comes next. The definition of $JI$ also adds an $r$ in front of this term that should not be there.

And finally, for axisymmetric flow, the entire equation is divided by the local radius $r$. This adds the $1/r$ coefficient where it should be added, and removes the $r$ in front of the $\Delta \hat{E}_r$ terms.

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. For sections of the code that apply to both sweeps (i.e., the implicit terms and the division by $r$ at the end), the first two subscripts are written as $(IV, I)$. For sections of the code that apply only to the first sweep, the first two subscripts are written as $(I2, I)$. For sections that apply only to the second sweep, they are written as $(I1, I2)$. The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.
3. The coding of the extra coefficients and source terms in the axisymmetric form of the equations is separate from the rest of the coding, and is bypassed if the flow is not axisymmetric. Similarly, the coding of coefficients and source terms involving the swirl velocity is separate from the rest of the coding, and is bypassed if there is no swirl.
4. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous and heat conduction terms.
5. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous and heat conduction terms containing derivatives in the specified direction.
Subroutine COEFS1 (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECT</td>
<td></td>
<td>Compute coefficients and source terms for the ( k ) and ( \varepsilon ) equations for the first ADI sweep.</td>
</tr>
</tbody>
</table>

**Input**

- **CMUR**
  - Constant \( C_{\mu} \) in formula for \( C_\mu \).
- **CTHREE**
  - Constant \( C_3 \) in formula for \( C_u \).
- **CTWOR**
  - Constant \( C_2 \) in formula for \( C_\varepsilon \).
- **DTAU**
  - Time step \( \Delta \tau \).
- **Dummy**
  - Distance to the nearest solid wall.
- **DXI, DETA**
  - Computational grid spacing \( \Delta \xi \) and \( \Delta \eta \).
- **E, EL**
  - Turbulent dissipation rate \( \varepsilon \) at time levels \( n \) and \( n-1 \).
- **ETAX, ETAY**
  - Metric coefficients \( \eta_x \) and \( \eta_y \) (or \( \eta_r \) if axisymmetric).
- **IAXI**
  - Flag for axisymmetric flow.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, \( J^{-1} \).
- **KE, KEL**
  - Turbulent kinetic energy \( k \) at time levels \( n \) and \( n-1 \).
- **MU**
  - Laminar viscosity \( \mu \) at time level \( n \).
- **MUT, MUTL**
  - Turbulent viscosity \( \mu_\tau \) at time levels \( n \) and \( n-1 \).
- **NPTS**
  - Number of grid points in the sweep direction, \( N \).
- **NV**
  - Number of grid points in the "vectorized" direction, \( N_v \).
- **NVD, NPTSD**
  - Leading two dimensions for the arrays \( A, B, C, \) and \( S \).
- **RER**
  - Reference Reynolds number \( Re_\rho \).
- **RHO, U, V**
  - Static density \( \rho \), and velocities \( u \) and \( v \), at time level \( n \).
- **RHOL**
  - Static density \( \rho \) at time level \( n-1 \).
- **SIGE, SIGK**
  - Constants \( \sigma_k \) and \( \sigma_\varepsilon \) used in the diffusion term of the \( \varepsilon \) equation.
- **TFAC**
  - Factor used in computing the \( k-\varepsilon \) time step.
- **THKE**
  - Parameters \( \theta_1 \) and \( \theta_2 \) determining type of time differencing for the \( k-\varepsilon \) equations.
- **VORT**
  - Production rate of turbulent kinetic energy.
- **XIX, XIY**
  - Metric coefficients \( \xi_x \) and \( \xi_y \) (or \( \xi_r \) if axisymmetric).
- **Y**
  - Radial coordinate \( r \) for axisymmetric flow.
- **YPLUSD**
  - Nondimensional distance \( y^* \) from the nearest solid wall.

**Output**

- **A, B, C**
  - Coefficient submatrices \( A, B, \) and \( C \) at interior points.
- **S**
  - Source term subvector \( S \) at interior points.
Description

Subroutine COEFS1 computes the coefficients and source terms for the $k$-$\epsilon$ equations for the first ADI sweep. Equation (9.40a) in Volume 1 represents, in vector form, the governing equation for the first ADI sweep for 2-D planar flow. This equation may be written in difference form as:

$$
\begin{bmatrix}
1 - T_1 Z_{11} & - T_1 Z_{12} \\
- T_1 Z_{21} & 1 - T_1 Z_{22}
\end{bmatrix}
\begin{bmatrix}
\Delta \hat{W}_1 \\
\Delta \hat{W}_2
\end{bmatrix}
+ T_1 \left\{ \left( \delta_u \right)_\xi \left( \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \Delta \hat{W}_1 \\ \Delta \hat{W}_2 \end{bmatrix} \right) - \delta_\xi \left( \begin{bmatrix} B_{11} & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} \Delta \hat{W}_1 \\ \Delta \hat{W}_2 \end{bmatrix} \right) \right\} = 
$$

\[
T_1 \left( \delta_\xi \hat{F}_M^\eta - \delta_\xi \hat{F}_M^\eta - 1 + \delta_\eta \hat{G}_M^\eta - \delta_\eta \hat{G}_M^\eta \right) 
+ T_2 \left( - (\delta_u)_\xi \hat{F}_C^\eta + \delta_\xi \hat{F}_D^\eta + \delta_\xi \hat{F}_M^\eta - (\delta_u)_\eta \hat{G}_C^\eta + \delta_\eta \hat{G}_D^\eta + \delta_\eta \hat{G}_M^\eta + \hat{S}^\eta + \hat{T}^\eta \right) + T_3 \hat{W}^\eta - 1
\]

where

\[
T_1 = \frac{\theta_1 \Delta \tau}{1 + \theta_2}
\]

\[
T_2 = \frac{\Delta \tau}{1 + \theta_2}
\]

\[
T_3 = \frac{\theta_2}{1 + \theta_2}
\]

In the above equation, $Z_{ij}$, etc., are elements of a matrix $Z$, defined as $Z = M + N$, and $A$, $B$, $M$, and $N$ are the Jacobian coefficient matrices defined in equations (9.31), (9.32), (9.35), and (9.36) of Volume 1. Also, $(\delta_u)_\xi$ is the first-order upwind difference operator used for the convective terms, and $\delta_\xi$ is the second-order central difference operator used for the viscous terms.

The convective term on the left side can thus be expanded as:

\[
(\delta_\xi)_\xi \left( \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \Delta \hat{W}_1 \\ \Delta \hat{W}_2 \end{bmatrix} \right) = \begin{cases} 
\frac{1}{\Delta \xi} \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \Delta \hat{W}_1 \\ \Delta \hat{W}_2 \end{bmatrix} & \text{if } \xi, \nu > 0 \\
\frac{1}{\Delta \xi} \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \Delta \hat{W}_1 \\ \Delta \hat{W}_2 \end{bmatrix} & \text{if } \xi, \nu > 0
\end{cases}
\]

In the above equations, and in those to follow, the subscripts $i$ and $j$ represent grid point indices in the $\xi$ and $\eta$ directions. For notational convenience, terms without an explicitly written $i$ or $j$ subscript are understood to be at $i$ or $j$.

The Jacobian coefficient matrix $B$ may be written as:

\[
B = \begin{bmatrix} f_9_{\xi(1)} & 0 \\
0 & f_9_{\xi(2)} \end{bmatrix}
\]

The viscous term on the left hand side may thus be expanded as:
\[
\delta \left( \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix} \Delta \hat{W}_1 \right) = \frac{1}{2(\Delta z)} \left( \begin{bmatrix} U - f_1 \xi g_{i-1} + 1(11) \\ U - f_1 \xi g_{i-1} + 1(22) \end{bmatrix} \Delta \hat{W}_1 \right) - \left( \begin{bmatrix} U + f_1 \xi g_{i+1} + 1(11) \\ U + f_1 \xi g_{i+1} + 1(22) \end{bmatrix} \Delta \hat{W}_2 \right) \]

On the right hand side, the convective term \( (\delta_{\xi})_j \hat{F}_C \) is differenced as:

\[
(\delta_{\xi})_j \hat{F}_C = \begin{cases} \frac{1}{\Delta z^2} \left[ (\hat{F}_C)_i - (\hat{F}_C)_i+1 \right] & \text{if } \xi_x u + \xi_y v > 0 \\
\frac{1}{\Delta z^2} \left[ (\hat{F}_C)_i+1 - (\hat{F}_C)_i \right] & \text{if } \xi_x u + \xi_y v < 0 
\end{cases}
\]

An analogous expression may be written for \( (\delta_{\eta})_n \hat{G}_C \). The vectors \( \hat{F}_D \) and \( \hat{F}_M \) may be written as

\[
\hat{F}_D = \begin{bmatrix} f_{g_{11}}(1) \\ f_{g_{12}}(2) \end{bmatrix}, \quad \hat{F}_M = \begin{bmatrix} f_{g_{21}}(1) \\ f_{g_{22}}(2) \end{bmatrix}
\]

The terms \( \delta_t \hat{F}_D \) and \( \delta_t \hat{F}_M \) are thus differenced as:

\[
\delta_t \hat{F}_D = \frac{1}{2(\Delta z)} \left\{ \begin{bmatrix} U - f_1 \xi g_{i-1} + 1(11) \\ U - f_1 \xi g_{i-1} + 1(22) \end{bmatrix} - \begin{bmatrix} U + f_1 \xi g_{i+1} + 1(11) \\ U + f_1 \xi g_{i+1} + 1(22) \end{bmatrix} \right\}
\]

Analogous expressions may be written for \( \delta_t \hat{G}_D \) and \( \delta_t \hat{G}_M \).

The \( k-e \) equations for axisymmetric flow are presented in Appendix B of Volume 1. They may be written in difference form for the first ADI sweep as:

\[
\begin{bmatrix} 1 - T_1 Z_{11} & - T_1 Z_{12} \\ - T_1 Z_{21} & 1 - T_1 Z_{22} \end{bmatrix} \Delta \hat{W}_1 + \frac{T_1}{r} (\delta_t)_t \left( \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} \Delta \hat{W}_1 \right) - \delta_t \left( \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix} \Delta \hat{W}_2 \right) = \frac{T_1}{r} \left( \delta_t \hat{F}_M - \delta_t \hat{F}_D - \delta_t \hat{G}_D - \delta_t \hat{G}_M \right)
\]

Proteus 2-D Programmer's Reference

4.0 Proteus Subprograms: COEFSI 153
where \( r \) is the radial coordinate and all other terms are the same as the 2-D planar equations presented above.

**Remarks**

1. For the variables \( A, B, C, \) and \( S \), the first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. Since this subroutine only applies to the first sweep, the first two subscripts of \( A, B, C, \) and \( S \) variables are written as \((J,I)\). The third subscript on \( A, B, C, \) and \( S \) corresponds to the equation. And, for \( A, B, \) and \( C \), the fourth subscript corresponds to the dependent variable for which \( A, B, \) or \( C \) is a coefficient.

2. For axisymmetric flows, the Fortran variables \( R_{IJ}, R_{I+1J}, \) and \( R_{I-1J} \) are the cylindrical \( r \) coordinates for the grid points \((I,J), (I+1,J), \) and \((I-1,J)\), respectively. Similarly, \( R_{I+1J} \) and \( R_{I-1J} \) are the cylindrical \( r \) coordinates for the grid points \((I,J+1)\) and \((I,J-1)\). For 2-D flows, all of these variables are set equal to 1.0.
Subroutine COEFS2 (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECT</td>
<td></td>
<td>Compute coefficients and source terms for the k and ( \varepsilon ) equations for the second ADI sweep.</td>
</tr>
</tbody>
</table>

**Input**

- **DETA**
  - Computational grid spacing \( \Delta \eta \).
- **DTAU**
  - Time step \( \Delta \tau \).
- **E, EL**
  - Turbulent dissipation rate \( \varepsilon \) at time levels \( n \) and \( n - 1 \).
- **ETAX, ETAY**
  - Metric coefficients \( \eta_x \) and \( \eta_y \) (or \( \eta \), if axisymmetric).
- **IAXI**
  - Flag for axisymmetric flow.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, \( J^{-1} \).
- **KE, KEL**
  - Turbulent kinetic energy \( k \) at time levels \( n \) and \( n - 1 \).
- **MU**
  - Laminar viscosity \( \mu \), at time level \( n \).
- **MUT**
  - Turbulent viscosity \( \mu_t \), at time level \( n \).
- **NPTS**
  - Number of grid points in the sweep direction, \( N \).
- **NV**
  - Number of grid points in the "vectorized" direction, \( N_v \).
- **NVD, NPTSD**
  - Leading two dimensions for the arrays A, B, C, and S.
- **RER**
  - Reference Reynolds number \( Re_\infty \).
- **RHO, U, V**
  - Static density \( \rho \), and velocities \( u \) and \( v \), at time level \( n \).
- **RHOL**
  - Static density \( \rho \) at time level \( n - 1 \).
- **S**
  - Computed solution subvector from first sweep.
- **SIGE, SIGK**
  - Constants \( \sigma_1 \) and \( \sigma_2 \) used in the diffusion term of the \( \varepsilon \) equation.
- **TFACT**
  - Factor used in computing the \( k-\varepsilon \) time step.
- **THKE**
  - Parameters \( \delta_1 \) and \( \delta_2 \) determining type of time differencing for the \( k-\varepsilon \) equations.
- **Y**
  - Radial coordinate \( r \) for axisymmetric flow.

**Output**

- **A, B, C**
  - Coefficient submatrices A, B, and C at interior points.
- **S**
  - Source term subvector S at interior points.

**Description**

Subroutine COEFS2 computes the coefficients and source terms for the \( k-\varepsilon \) equations for the second ADI sweep. Equation (9.40b) in Volume 1 represents, in vector form, the governing equation for the second ADI sweep for 2-D planar flow. This equation may be written in difference form as:

\[
\begin{bmatrix}
\Delta \hat{W}_1^n \\
\Delta \hat{W}_2^n
\end{bmatrix} + T_1 \begin{bmatrix}
(\delta \eta)_{n} \begin{bmatrix}
C_{11} & 0 \\
0 & C_{22}
\end{bmatrix} \begin{bmatrix}
\Delta \hat{W}_1^n \\
\Delta \hat{W}_2^n
\end{bmatrix}
\end{bmatrix} - \delta \eta \begin{bmatrix}
D_{11} & 0 \\
0 & D_{22}
\end{bmatrix} \begin{bmatrix}
\Delta \hat{W}_1^n \\
\Delta \hat{W}_2^n
\end{bmatrix} = \begin{bmatrix}
\Delta \hat{W}_1^* \\
\Delta \hat{W}_2^*
\end{bmatrix}
\]

Proteus 2-D Programmer's Reference 4.0 Proteus Subprograms: COEFS2 155
where

\[ T_1 = \frac{\theta_1 \Delta \tau}{1 + \theta_2} \]

In the above equation \( C \) and \( D \) are the Jacobian coefficient matrices defined in equations (9.33) and (9.34) of Volume 1. Also, \((\delta_\eta)_q\) is the first-order upwind difference operator used for the convective terms, and \(\delta_r\) is the second-order central difference operator used for the viscous terms.

The convective term on the left side can thus be expanded as:

\[
(\delta_\eta)_q \left( \begin{bmatrix} C_{11} & 0 & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \\ 0 & C_{22} & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \end{bmatrix} \right) = \begin{cases} \frac{-1}{\Delta \eta} \left[ \begin{bmatrix} C_{11} & 0 \\ 0 & C_{22} \end{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \right] - \frac{1}{\Delta \eta} \left[ \begin{bmatrix} C_{11} & 0 \\ 0 & C_{22} \end{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \right]_{j+1} & \text{if } \eta_{i,j} + \eta_{i,j+1} > 0 \\
\frac{-1}{\Delta \eta} \left[ \begin{bmatrix} C_{11} & 0 \\ 0 & C_{22} \end{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \right] + \frac{1}{\Delta \eta} \left[ \begin{bmatrix} C_{11} & 0 \\ 0 & C_{22} \end{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \right]_{j-1} & \text{if } \eta_{i,j} + \eta_{i,j+1} < 0 
\end{cases}
\]

In the above equations, and in those to follow, the subscripts \( i \) and \( j \) represent grid point indices in the \( \xi \) and \( \eta \) directions. For notational convenience, terms without an explicitly written \( i \) or \( j \) subscript are understood to be at \( i \) or \( j \).

The Jacobian coefficient matrix \( D \) may be written as:

\[
D = \begin{bmatrix} f_{\eta(i1)} & 0 \\ 0 & f_{\eta(j2)} \end{bmatrix}
\]

The viscous term on the left hand side may thus be expanded as:

\[
\delta_\eta \left( \begin{bmatrix} D_{11} & 0 & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \\ 0 & D_{22} & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \end{bmatrix} \right) = \frac{1}{2(\Delta \xi)^2} \left[ \begin{bmatrix} (f_{j-1} + f_j)g_{j-1}(11) & 0 \\ 0 & (f_{j-1} + f_j)g_{j-1}(22) \end{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \right]_{j-1} - \frac{1}{2(\Delta \xi)^2} \left[ \begin{bmatrix} (f_{j+1} + 2f_j + f_{j-1})g_{j}(11) & 0 \\ 0 & (f_{j+1} + 2f_j + f_{j-1})g_{j}(22) \end{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \right]_{j} + \frac{1}{2(\Delta \xi)^2} \left[ \begin{bmatrix} (f_{j+1} + f_j + 1)g_{j+1}(11) & 0 \\ 0 & (f_{j+1} + f_j + 1)g_{j+1}(22) \end{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \right]_{j+1}
\]

The \( k-\epsilon \) equations for axisymmetric flow are presented in Appendix B of Volume 1. They may be written in difference form for the second ADI sweep as:

\[
\begin{bmatrix} \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \\ \frac{\Delta \hat{W}_2}{\Delta \hat{W}_2} \end{bmatrix} + \frac{T_1}{r} \left( \delta_\eta \left( \begin{bmatrix} C_{11} & 0 & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \\ 0 & C_{22} & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \end{bmatrix} \right) - \delta_r \left( \begin{bmatrix} D_{11} & 0 & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \\ 0 & D_{22} & \frac{\Delta \hat{W}_1}{\Delta \hat{W}_2} \end{bmatrix} \right) \right) = \begin{bmatrix} \Delta \hat{W}_1^* \\ \Delta \hat{W}_2^* \end{bmatrix}
\]

where \( r \) is the radial coordinate and all other terms are the same as the 2-D planar equations presented above.

**Remarks**

1. For the variables \( A, B, C \), and \( S \), the first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. Since this subroutine only ap-
plies to the second sweep, the first two subscripts of A, B, C, and S variables are written as (I,J). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.

2. For axisymmetric flows, the Fortran variables RIJ, RJP1, and RIJM1 are the cylindrical \( r \) coordinates for the grid points (I,J), (I,J + 1), and (I,J - 1), respectively. For 2-D flows, these variables are set equal to 1.0.
Subroutine COEFX (A,B,C,METX,METY,METT,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td></td>
<td>Compute coefficients and source term for the (x)-momentum equation.</td>
</tr>
</tbody>
</table>

### Input

- **DEL**
  - Computational grid spacing in sweep direction.
- **DPDRHO, DPDRU, DPDRV, DPDRW, DPDET**
  - Derivatives \(\partial p / \partial \rho, \partial p / \partial (\rho u), \partial p / \partial (\rho v), \partial p / \partial (\rho w)\), and \(\partial p / \partial E_r\).
- **DTAU**
  - Time step \(\Delta t\).
- **DXI, DETA**
  - Computational grid spacing \(\Delta \xi\) and \(\Delta \eta\).
- **ETAX, ETAY, ETAT**
  - Metric coefficients \(\eta_x, \eta_y\) (or \(\eta_y\) if axisymmetric), and \(\eta_t\).
- **IAXI**
  - Flag for axisymmetric flow.
- **IBASE, ISTEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IEULER**
  - Flag for Euler calculation.
- **IHSTAG**
  - Flag for constant stagnation enthalpy option.
- **ISWEEP**
  - Current ADI sweep number.
- **ISWIRL**
  - Flag for swirl in axisymmetric flow.
- **ITHIN**
  - Flags for thin-layer option.
- **IV**
  - Index in the "vectorized" direction, \(i\).
- **I1, I2**
  - Grid indices \(i\) and \(j\), in the \(\xi\) and \(\eta\) directions.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, \(J^{-1}\) (times the radius \(r\) for axisymmetric flow.)
- **METX, METY, METT**
  - Derivatives of sweep direction computational coordinate with respect to \(x\), \(y\) (or \(r\) if axisymmetric), and \(t\).
- **MU, LA**
  - Effective coefficient of viscosity \(\mu\) and effective second coefficient of viscosity \(\lambda\) at time level \(n\).
- **NEQ**
  - Number of coupled equations being solved, \(N_{eq}\).
- **NPTS**
  - Number of grid points in the sweep direction, \(N\).
- **NR, NRU, NRV, NRW, NET**
  - Array indices associated with the dependent variables \(\rho, \rho u, \rho v, \rho w,\) and \(E_r\).
- **NVD, NPTSD**
  - Leading two dimensions for the arrays \(A, B, C, S, METX, METY,\) and \(METT\).
- **NXM**
  - Array index associated with the \(x\)-momentum equation.
- **P**
  - Static pressure \(p\) at time level \(n\).
- **RAX**
  - \(1\) for two-dimensional planar flow, and the local radius \(r\) for axisymmetric flow.
- **RER**
  - Reference Reynolds number \(Re\).
- **RHO, U, V**
  - Static density \(\rho\), and velocities \(u\) and \(v\) at time level \(n\).
- **RHOL, UL, VL**
  - Static density \(\rho\), and velocities \(u\) and \(v\) from previous ADI sweep.
**THX**

Parameters $\theta_1$, $\theta_2$, and $\theta_3$ determining type of time differencing for the $x$-momentum equation.

**XIX, XIY, XIT**

Metric coefficients $\xi_x$, $\xi_y$ (or $\xi_r$ if axisymmetric), and $\xi_r$.

Radial coordinate $r$ for axisymmetric flow.

**Output**

A, B, C

Coefficient submatrices A, B, and C at interior points (row NXM only).

S

Source term subvector S at interior points (element NXM only).

**Description**

Subroutine COEFX computes the coefficients and source term for the $x$-momentum equation. Equations (7.5a-b) in Volume 1 represent, in vector form, the four governing difference equations for the two ADI sweeps for 2-D planar flow. The elements of the inviscid flux vectors $\mathbf{E}$ and $\mathbf{F}$ are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors $\mathbf{E}_{y_1}$, $\mathbf{E}_{y_2}$, etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices $\partial \mathbf{E} / \partial \mathbf{Q}$, $\partial \mathbf{E}_{y_1} / \partial \mathbf{Q}$, etc., are given in Section 4.0 of Volume 1.

Using all of these equations, the differenced form of the $x$-momentum equation for 2-D planar flow may be written for the two ADI sweeps as

**Sweep 1 ($\xi$ direction)**

$$
\Delta (\hat{\rho} \hat{u})_i^n + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2 \Delta \xi} \left[ \left( \frac{\partial \hat{E}_2}{\partial \hat{Q}} \right)^n_{i+1} - \left( \frac{\partial \hat{E}_2}{\partial \hat{Q}} \right)^n_{i-1} \right] - \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2 \Delta \xi} \left[ (f_i - 1 + f_i)^n g_{i-1} \Delta \hat{Q}_{i-1}^n - (f_i - 1 + 2f_i + f_i + 1)^n g_i \Delta \hat{Q}_i^n + (f_i + f_i + 1)^n g_{i+1} \Delta \hat{Q}_{i+1}^n \right] =$$

$$- \frac{\Delta \tau}{1 + \theta_2} \left( \delta_i \hat{E}_2 + \delta_i \hat{F}_2 \right)^n + \frac{\Delta \tau}{1 + \theta_2} \left[ \delta_i \left( \hat{E}_{y_1} \right)_{2} + \delta_i \left( \hat{F}_{y_2} \right)_{2} \right]^n + \frac{(1 + \theta_3) \Delta \tau}{1 + \theta_2} \left[ \delta_i \left( \hat{E}_{y_2} \right)_{2} + \delta_i \left( \hat{F}_{y_2} \right)_{2} \right]^n - \frac{\theta_3 \Delta \tau}{1 + \theta_2} \left[ \delta_i \left( \hat{E}_{y_2} \right)_{2} + \delta_i \left( \hat{F}_{y_2} \right)_{2} \right] + \frac{\theta_2}{1 + \theta_2} \Delta (\hat{\rho} \hat{u})_{i}^{n-1}
$$

**Sweep 2 ($\eta$ direction)**

$$
\Delta (\hat{\rho} \hat{u})_j^n + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2 \Delta \eta} \left[ \left( \frac{\partial \hat{F}_2}{\partial \hat{Q}} \right)^n_{j+1} - \left( \frac{\partial \hat{F}_2}{\partial \hat{Q}} \right)^n_{j-1} \right] - \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2 \Delta \eta} \left[ (f_j - 1 + f_j)^n g_{j-1} \Delta \hat{Q}_{j-1}^n - (f_j - 1 + 2f_j + f_j + 1)^n g_j \Delta \hat{Q}_j^n + (f_j + f_j + 1)^n g_{j+1} \Delta \hat{Q}_{j+1}^n \right] =$$

$$\Delta (\hat{\rho} \hat{u})_{j}^{n-1}
$$

---

20 These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy $E_T$ would not appear as a dependent variable, and the Jacobian coefficient matrices would have only three elements.
In the above equations, the subscripts $i$ and $j$ represent grid point indices in the $\xi$ and $\eta$ directions. For notational convenience, terms without an explicitly written $i$ or $j$ subscript are understood to be at $i$ or $j$.

On the left hand side, $f$ is the coefficient of $\partial/\partial \xi$ (or $\partial/\partial \eta$, depending on the sweep) in the $\partial \hat{E}_{vi}/\partial \hat{Q}$ (or $\partial \hat{F}_{vi}/\partial \hat{Q}$) Jacobian coefficient matrix. Similarly, $g$ is the term in the parentheses following $\partial/\partial \xi$ (or $\partial/\partial \eta$) in the $\partial \hat{E}_{vi}/\partial \hat{Q}$ (or $\partial \hat{F}_{vi}/\partial \hat{Q}$) Jacobian coefficient matrix.

The vector of dependent variables is

$$\hat{Q} = \frac{1}{J} [\rho \quad \rho u \quad \rho v \quad E_T]^T$$

The appropriate elements of the inviscid flux vectors are given by

$$\hat{E}_2 = \frac{1}{J} \left[ \left( \rho u^2 + p \right) \xi_x + \rho \omega_2 \xi_y + \rho u \xi_z \right]$$

$$\hat{F}_2 = \frac{1}{J} \left[ \left( \rho u^2 + p \right) \eta_x + \rho \omega_2 \eta_y + \rho u \eta_z \right]$$

The appropriate elements of the non-cross derivative viscous flux vectors are

$$\left( \hat{E}_{vi} \right)_2 = \frac{1}{J} \left[ 2 \mu \xi_x \eta_x \xi_x^2 + \lambda \xi_x (\xi_x \eta_x + \xi_y \eta_y) + \mu \xi_y (\xi_y \mu_x + \xi_x \eta_y) \right]$$

$$\left( \hat{F}_{vi} \right)_2 = \frac{1}{J} \left[ 2 \mu \eta_x \xi_x \eta_x^2 + \lambda \eta_x (\xi_x \eta_x + \xi_y \eta_y) + \mu \eta_y (\eta_y \mu_x + \eta_x \eta_y) \right]$$

And the appropriate elements of the cross derivative viscous flux vectors are

$$\left( \hat{E}_{vi} \right)_2 = \frac{1}{J} \left[ 2 \mu \xi_x \eta_x \xi_x^2 + \lambda \xi_x (\xi_x \eta_x + \xi_y \eta_y) + \mu \xi_y (\xi_y \mu_x + \xi_x \eta_y) \right]$$

$$\left( \hat{F}_{vi} \right)_2 = \frac{1}{J} \left[ 2 \mu \eta_x \xi_x \eta_x^2 + \lambda \eta_x (\xi_x \eta_x + \xi_y \eta_y) + \mu \eta_y (\eta_y \mu_x + \eta_x \eta_y) \right]$$

The elements of the Jacobian coefficient matrix $\partial \hat{E}_i/\partial \hat{Q}$ for the inviscid terms in the $x$-momentum equation are

$$\frac{\partial \hat{E}_2}{\partial \hat{Q}} = \left[ \frac{\partial p}{\partial \rho} \xi_x - uf_1 \xi_x + f_1 + u^2 \xi_x + \frac{\partial p}{\partial (\rho u)} \xi_x + u \xi_y + \frac{\partial p}{\partial (\rho v)} \xi_x \right]$$

where $f_i = u \xi_x + v \xi_y$.

The elements of the Jacobian coefficient matrix $\partial \hat{E}_{vi}/\partial \hat{Q}$ for the viscous terms are

$$\frac{\partial \left( \hat{E}_{vi} \right)_2}{\partial \hat{Q}} = \frac{1}{Re} \left[ \left( \begin{array}{c} \frac{\partial \hat{E}_1}{\partial \hat{Q}} \\ \frac{\partial \hat{E}_2}{\partial \hat{Q}} \end{array} \right) \begin{array}{ccc} \frac{\partial}{\partial \xi} & \frac{\partial}{\partial \eta} \\ \frac{1}{\rho} & \frac{1}{\rho} \end{array} \right]$$

where

$$\left( \frac{\partial \hat{E}_1}{\partial \hat{Q}} \right)_2 \begin{array}{c} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{array} = \begin{array}{c} \alpha_{xx} \frac{\partial}{\partial \xi} \left( \frac{u}{\rho} \right) - \alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{v}{\rho} \right) \\ \alpha_{yy} \frac{\partial}{\partial \eta} \left( \frac{u}{\rho} \right) - \alpha_{yx} \frac{\partial}{\partial \eta} \left( \frac{v}{\rho} \right) \end{array}$$
\[ \alpha_{xx} = (2\mu + \lambda) \xi^2_x + \mu \xi^2_y \]
\[ \alpha_{xy} = (\mu + \lambda) \xi_x \xi_y \]

The Jacobian coefficient matrices \( \partial \hat{F}_z / \partial \hat{Q} \) and \( \partial (\hat{F}_V)_z / \partial \hat{Q} \) have the same form as \( \partial \hat{E}_z / \partial \hat{Q} \) and \( \partial (\hat{E}_V)_z / \partial \hat{Q} \), but with \( \xi \) replaced by \( \eta \).

As an example of how these equations are translated into Fortran, consider the \( \Delta(\rho u/j) \) term on the left hand side for the first sweep. This is the second element of \( \hat{Q} \), so using the second element in \( \partial \hat{E}_z / \partial \hat{Q} \), and including the \( \Delta(\rho u/j) \) term, we get for the inviscid term

\[
A(IV,I,NXM,NRU) = -\frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left[ (\xi)_{i-1,j} + (u\xi_x + v\xi_y)_{i-1,j} + (u\xi_x)_{i-1,j} + \left( \frac{\partial \rho}{\partial (\rho u)} \xi_x \right)_{i-1,j} \right]
\]

\[
B(IV,I,NXM,NRU) = 1
\]

\[
C(IV,I,NXM,NRU) = \frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left[ (\xi)_{i+1,j} + (u\xi_x + v\xi_y)_{i+1,j} + (u\xi_x)_{i+1,j} + \left( \frac{\partial \rho}{\partial (\rho u)} \xi_x \right)_{i+1,j} \right]
\]

For the viscous terms on the left hand side, we use the second element in \( \partial (\hat{E}_V)_z / \partial \hat{Q} \), which is

\[
\frac{1}{Re} \alpha_{xx} \frac{\partial}{\partial \xi} \left( \frac{1}{\rho} \right)
\]

Thus \( f = \alpha_{xx} / Re \) and \( g = 1/\rho \). To add the viscous contribution to this part of the A coefficient submatrix, we therefore set

\[
A(IV,I,NXM,NRU) = A(IV,I,NXM,NRU) - \frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left[ (\alpha_{xx})_{i-1,j} + (\alpha_{xx})_{i,j} \right] \left( \frac{1}{\rho} \right)_{i-1,j}
\]

Similar equations may be written for the B and C coefficient submatrices.

The equations for axisymmetric flow are developed in Appendix B of Volume 1. The axisymmetric \( x \)-momentum equation for the two ADI sweeps is given by\(^{21}\)

**Sweep 1 (\( \xi \) direction)**

\[
\Delta(\rho u)^*_i = \frac{\theta_1(\Delta \tau)}{(1 + \theta_2)2\Delta \xi} \left[ \left( \frac{\partial \hat{E}_z}{\partial \hat{Q}} \right)_{i+1}^* \delta \hat{Q}^*_{i+1} - \left( \frac{\partial \hat{E}_z}{\partial \hat{Q}} \right)_{i-1}^* \delta \hat{Q}^*_{i-1} \right]
\]

\[
- \frac{\theta_1(\Delta \tau)}{(1 + \theta_2)2(\Delta \xi)^2} \left[ \delta \left[ \left( \frac{\partial \hat{E}_z}{\partial \hat{E}_v} + \frac{\partial \hat{E}_z}{\partial \hat{E}_{V,2}} \right)_{r,\xi} + \frac{\Delta \tau}{1 + \theta_2} \left( \delta \hat{E}_z(\hat{E}_V)_{v,2} + \delta \hat{E}_z(\hat{E}_{V,2})_{v,2} \right) \right]_r \right]
\]

\[
+ \frac{\Delta \tau}{1 + \theta_2} \left[ \delta \left[ \left( \frac{\partial \hat{E}_z}{\partial \hat{E}_v} + \frac{\partial \hat{E}_z}{\partial \hat{E}_{V,2}} \right)_{r,\xi} + \frac{\Delta \tau}{1 + \theta_2} \left( \delta \hat{E}_z(\hat{E}_V)_{v,2} + \delta \hat{E}_z(\hat{E}_{V,2})_{v,2} \right) \right]_r \right]
\]

\[
- \frac{\theta_2(\Delta \tau)}{1 + \theta_2} \left( \delta \hat{E}_z(\hat{E}_V)_{v,2} + \delta \hat{E}_z(\hat{E}_{V,2})_{v,2} \right)_v - \frac{\theta_2(\Delta \tau)}{1 + \theta_2} \left( \delta \hat{E}_z(\hat{E}_V)_{v,2} + \delta \hat{E}_z(\hat{E}_{V,2})_{v,2} \right)_{v-1}^* + \frac{\theta_2(\Delta \tau)}{1 + \theta_2} \Delta(\rho u)^{\text{viscous}}
\]

\(^{21}\) These equations are written for the general case with swirl. For a non-swirl case, the swirl momentum \( \rho_w \) would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.
Sweep 2 ($\eta$ direction)

\[
\Delta(\hat{\omega})^n + \frac{\theta_1 \Delta t}{(1 + \theta_2)2\Delta \eta} \left[ \left( \frac{\partial \hat{\omega}}{\partial \tilde{Q}} \right)_{j+1}^{\eta} - \left( \frac{\partial \hat{\omega}}{\partial \tilde{Q}} \right)_{j-1}^{\eta} \right] \\
= -\frac{\theta_1 \Delta t}{(1 + \theta_2)2(\Delta \eta)^2} \left[ (\eta_{j-1} - 1 + \tau_1 f_1)^{\eta_{j-1}} - (\eta_{j-1} - 1 + 2r_1 f_1 + r_1 f_2) \hat{\omega}_{j-1}^{\eta} + (\eta_{j-1} + 1 + f_1) \hat{\omega}_{j+1}^{\eta} \right] = \Delta(\hat{\omega})^n
\]

where now

\[
\hat{Q} = \frac{1}{J} \begin{bmatrix} \rho & \rho u & \rho v & \rho w & E_T \end{bmatrix}^T
\]

\[
\hat{E}_2 = \frac{1}{J} \left[ (\rho u^2 + p)\xi_x + \rho \nu \xi_r + \rho \nu \xi_l \right]
\]

\[
\hat{F}_2 = \frac{1}{J} \left[ (\rho u^2 + p)n_x + \rho \nu n_r + \rho \nu n_l \right]
\]

\[
\begin{align*}
(\hat{E}_{V_1})_2 &= \frac{1}{J \text{Re}_t} \left[ 2\mu^2 \xi_x u \xi_x + \lambda \xi_x \left( \xi_r u \xi_r + \frac{1}{r} \xi_r \right) \xi_x + \mu \xi_r (\xi_r u \xi_r + \xi_r \xi_x) \right] \\
(\hat{F}_{V_1})_2 &= \frac{1}{J \text{Re}_t} \left[ 2\mu^2 \xi_x n \xi_x + \lambda \xi_x \left( n_r \eta \xi_r + \frac{1}{r} \eta \right) \xi_x + \mu \xi_r (n_r \eta \xi_r + n_r \xi_x) \right] \\
(\hat{E}_{V_2})_2 &= \frac{1}{J \text{Re}_t} \left[ 2\mu^2 \xi_x n \xi_x + \lambda \xi_x \left( n_r \eta \xi_r + \frac{1}{r} \eta \right) \xi_x + \mu \xi_r (n_r \eta \xi_r + n_r \xi_x) \right] \\
(\hat{F}_{V_2})_2 &= \frac{1}{J \text{Re}_t} \left[ 2\mu^2 \xi_x n \xi_x + \lambda \xi_x \left( n_r \eta \xi_r + \frac{1}{r} \eta \right) \xi_x + \mu \xi_r (n_r \eta \xi_r + n_r \xi_x) \right]
\end{align*}
\]

The elements of the Jacobian coefficient matrix $\partial \hat{E}_2 / \partial \hat{Q}$ for the inviscid terms in the axisymmetric form of the energy equation are

\[
\frac{\partial \hat{E}_2}{\partial \hat{Q}} = \begin{bmatrix}
\frac{\partial \rho}{\partial \hat{Q}} & \xi_x - u \xi_x + f_i + u \xi_x + \frac{\partial \rho}{\partial (\rho u)} & \xi_x & u \xi_r + \frac{\partial \rho}{\partial (\rho v)} & \xi_x & \frac{\partial \rho}{\partial (\rho w)} & \xi_x & \frac{\partial \rho}{\partial (\rho E_T)} & \xi_x
\end{bmatrix}
\]

where $f_i = u \xi_x + v \xi_r$.

The elements of the Jacobian coefficient matrix $\partial \hat{E}_{V_1} / \partial \hat{Q}$ for the viscous terms are

\[
\frac{\partial \hat{E}_{V_1}}{\partial \hat{Q}} = \frac{1}{\text{Re}_t} \left[ \left( \frac{\partial \hat{E}_{V_1}}{\partial \hat{Q}} \right)_{21}^{\eta} \right]
\]

where

\[
\left( \frac{\partial \hat{E}_{V_1}}{\partial \hat{Q}} \right)_{21}^{\eta} = a_{xx} \frac{\partial \xi_x}{\partial \xi_x} - a_{xr} \frac{\partial \xi_x}{\partial \xi_r} - a_{x\xi} \frac{\partial \xi_x}{\partial \xi_x} + a_{x\xi} \frac{\partial \xi_x}{\partial \xi_r}
\]

\[
a_{xx} = (2\mu + \lambda) \xi_x^2 + \mu \xi_r^2
\]

162 4.0 Proteus Subprograms: COEFX  Proteus 2-D Programmer's Reference
As in 2-D planar flow, the Jacobian coefficient matrices \( \frac{\partial \hat{F}_2}{\partial \hat{Q}} \) and \( \frac{\partial \hat{F}_{\nu_1}}{\partial \hat{Q}} \) have the same form as \( \frac{\partial \hat{E}_2}{\partial \hat{Q}} \) and \( \frac{\partial \hat{E}_{\nu_1}}{\partial \hat{Q}} \), but with \( \xi \) replaced by \( \eta \).

Note that the equations for 2-D planar and axisymmetric flow are very similar. In the axisymmetric equations there are some additional terms involving the radius \( r \) in the viscous flux vectors, with corresponding terms in the Jacobian coefficient matrices. The radius \( r \) appears as an additional coefficient in front of the flux vectors \( \hat{E}, \hat{E}_{\nu_1} \), etc., and in front of the Jacobian coefficient matrices \( \frac{\partial \hat{E}_2}{\partial \hat{Q}}, \frac{\partial \hat{E}_{\nu_1}}{\partial \hat{Q}} \), etc. In addition, \( 1/r \) appears in front of every term in the equation except the \( \Delta (\rho u) \) terms. In Proteus, the Fortran variables are defined in such a way that, for many terms, the same coding can be used for both 2-D planar and axisymmetric flow. Unfortunately, this may make some of the coding a little confusing. It is hoped that this detailed description, when compared with the source listing, will help make things clear.

In COEFX, the coefficients of the left hand side, or implicit, terms are defined first. The implicit terms for the second ADI sweep have exactly the same form as for the first sweep, but with \( \xi \) replaced by \( \eta \). By defining \( \text{DEL, METX, METY, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for both sweeps. The variable \( \text{RAX is equal to 1 for 2-D planar flow, and the radius } r \text{ for axisymmetric flow. This adds the } r \text{ in front of the Jacobian coefficient matrices for axisymmetric flow, but has no effect for 2-D planar flow. The } 1/r \text{ coefficient in front of each term will be added later. In this section of code, the coefficient of } \Delta (\rho u) \text{ (part of } B(IV,1,NXM,NRU) \text{) is set equal to } r, \text{ not 1 as it should be. This will be corrected later.} \)

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms. These formulas are presented in Section 5.0 of Volume 1. For axisymmetric flow, the Fortran variable \( JI \), which is normally defined as \( 1/J \), is temporarily redefined as \( r/J \) before the COEF routines are called. This automatically accounts for the \( r \) coefficient in front of all the flux vectors in the source term. The \( 1/r \) coefficient in front of each term will be added later. This definition of \( JI \) adds an \( r \) in front of the \( \Delta (\rho u)^* \) term that should not be there. This will also be corrected later.

The coding for the source term for the second sweep, which consists only of \( \Delta (\rho u)^* \), comes next. The definition of \( JI \) also adds an \( r \) in front of this term that should not be there.

And finally, for axisymmetric flow, the entire equation is divided by the local radius \( r \). This adds the \( 1/r \) coefficient where it should be added, and removes the \( r \) in front of the \( \Delta (\rho u) \) terms.

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables A, B, C, and S may be confusing. The first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. For sections of the code that apply to both sweeps (i.e., the implicit terms and the division by \( r \) at the end), the first two subscripts are written as \( (IV,I) \). For sections of the code that apply only to the first sweep, the first two subscripts are written as \( (I2,I1) \). For sections that apply only to the second sweep, they are written as \( (I1,I2) \). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.
3. The coding of the extra coefficients and source terms in the axisymmetric form of the equations is separate from the rest of the coding, and is bypassed if the flow is not axisymmetric. Similarly, the coding of coefficients and source terms involving the swirl velocity is separate from the rest of the coding, and is bypassed if there is no swirl.
4. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous terms.
5. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous terms containing derivatives in the specified direction.
**Subroutine COEFY (A,B,C,S,METX,METY,METT,NVD,NPTSD)**

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td></td>
<td>Compute coefficients and source term for the y or r-momentum equation.</td>
</tr>
</tbody>
</table>

**Input**

- **DEL**
  - Computational grid spacing in sweep direction.
- **DPDRHO, DPDRU, DPDRV, DPDRW, DPDET**
  - Derivatives $\partial p/\partial \rho$, $\partial p/\partial (\rho u)$, $\partial p/\partial (\rho v)$, $\partial p/\partial (\rho w)$, and $\partial p/\partial E_r$.
- **DTAU**
  - Time step $\Delta t$.
- **DXI, DETA**
  - Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **ETAX, ETAY, ETAT**
  - Metric coefficients $\eta_x$, $\eta_y$ (or $\eta_r$ if axisymmetric), and $\eta_t$.
- **IAXI**
  - Flag for axisymmetric flow.
- **IBASE, ISTEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IEULER**
  - Flag for Euler calculation.
- **IHSTAG**
  - Flag for constant stagnation enthalpy option.
- **ISWEEP**
  - Current ADI sweep number.
- **ISWIRL**
  - Flag for swirl in axisymmetric flow.
- **ITHIN**
  - Flags for thin-layer option.
- **IV**
  - Index in the "vectorized" direction, $i$.
- **I1, I2**
  - Grid indices $i$ and $j$, in the $\xi$ and $\eta$ directions.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$ (times the radius $r$ for axisymmetric flow.)
- **METX, METY, METT**
  - Derivatives of sweep direction computational coordinate with respect to $x$, $y$ (or $r$ if axisymmetric), and $t$.
- **MU, LA**
  - Effective coefficient of viscosity $\mu$ and effective second coefficient of viscosity $\lambda$ at time level $n$.
- **NEQ**
  - Number of coupled equations being solved, $N_{eq}$.
- **NPTS**
  - Number of grid points in the sweep direction, $N$.
- **NR, NRU, NRV, NRW, NET**
  - Array indices associated with the dependent variables $\rho$, $\rho u$, $\rho v$, $\rho w$, and $E_r$.
- **NVD, NPTSD**
  - Leading two dimensions for the arrays $A$, $B$, $C$, $S$, METX, METY, and METT.
- **NYM**
  - Array index associated with the $y$-momentum (or $r$-momentum if axisymmetric) equation.
- **P**
  - Static pressure $p$ at time level $n$.
- **RAX**
  - 1 for two-dimensional planar flow, and the local radius $r$ for axisymmetric flow.
- **RER**
  - Reference Reynolds number $Re_r$.
- **RHO, U, V, W**
  - Static density $\rho$, and velocities $u$, $v$, and $w$, at time level $n$. 

Proteus 2-D Programmer's Reference 4.0 Proteus Subprograms: COEFY 165
RHOL, UL, VL
* THY
XIX, XIY, XIT
Y

Static density $\rho$, and velocities $u$ and $v$ from previous ADI sweep.

Parameters $\theta_1$, $\theta_2$, and $\theta_3$ determining type of time differencing for the $y$-momentum equation.

Metric coefficients $\xi$, $\zeta$ (or $\xi$, if axisymmetric), and $\xi_r$.

Radial coordinate $r$ for axisymmetric flow.

Output

A, B, C

Coefficient submatrices A, B, and C at interior points (row NYM only).

S

Source term subvector S at interior points (element NYM only).

Description

Subroutine COEFY computes the coefficients and source term for the $y$-momentum equation for 2-D planar flow, or the $r$-momentum equation for axisymmetric flow. Equations (7.5a-b) in Volume 1 represent, in vector form, the four governing difference equations for the two ADI sweeps for 2-D planar flow.

The elements of the inviscid flux vectors $\hat{E}$ and $\hat{F}$ are given in Section 2.0 of Volume 1, and the elements of the viscous flux vectors $\hat{E}_{\nu_1}$, $\hat{E}_{\nu_2}$, etc., are given in Appendix A of Volume 1. The Jacobian coefficient matrices $\partial \hat{E}/\partial \hat{Q}$, $\partial \hat{E}_{\nu_1}/\partial \hat{Q}$, etc., are given in Section 4.0 of Volume 1. Using all of these equations, the differenced form of the $y$-momentum equation for 2-D planar flow may be written for the two ADI sweeps as

\[
\Delta \hat{\rho} \nu_1^* + \frac{\theta_1 \Delta \tau}{(1 + \theta_2) 2 \Delta \xi} \left[ \left( \frac{\partial \hat{E}_3}{\partial \hat{Q}} \right)^n_{i+1} \Delta \hat{Q}_i^* - \left( \frac{\partial \hat{E}_3}{\partial \hat{Q}} \right)^n_{i-1} \Delta \hat{Q}_i^{*^*} \right] + \frac{\theta_1 \Delta \tau}{(1 + \theta_2) 2 (\Delta \xi)} \left[ (f_i - 1 + f_i) g_i^{-1} \Delta \hat{Q}_i^* - (f_i - 1 + 2f_i + f_{i+1}) g_i^n \Delta \hat{Q}_i + (f_i + f_{i+1}) g_i^n A \Delta \hat{Q}_i^{*^*} \right] =
\]

\[
- \frac{\Delta \tau}{1 + \theta_2} \left( \delta_x \hat{E}_3 + \delta_y \hat{F}_3 \right)^n + \frac{\Delta \tau}{1 + \theta_2} \left[ \delta_x \hat{E}_{\nu_1} + \delta_y \hat{F}_{\nu_1} \right]^n + \frac{(1 + \theta_3) \Delta \tau}{1 + \theta_2} \left[ \delta_x \hat{E}_{\nu_2} + \delta_y \hat{F}_{\nu_2} \right]^n \frac{\theta_3 \Delta \tau}{1 + \theta_2} \left[ \delta_x \hat{E}_{\nu_3} + \delta_y \hat{F}_{\nu_3} \right]^n - \frac{\theta_2}{1 + \theta_2} \Delta (\hat{\rho} \nu)_{i-1} \]
\]

These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy $\hat{E}_T$ would not appear as a dependent variable, and the Jacobian coefficient matrices would have only three elements.
Sweep 2 ($\eta$ direction)

\[
\Delta(\hat{\rho}v)_j^n + \frac{\theta_1\Delta\tau}{(1 + \theta_2)2\Delta\eta} \left[ \left( \frac{\partial F_3}{\partial \hat{Q}} \right)_{j+1} - \left( \frac{\partial F_3}{\partial \hat{Q}} \right)_{j-1} \right] + \frac{\theta_1\Delta\tau}{(1 + \theta_2)(2\Delta\eta)^2} \left[ (f_j - f_j)g_{j-1}^n \hat{Q}_{j-1}^n - (f_j + 2f_j + f_{j+1})g_{j+1}^n \hat{Q}_{j+1}^n + (f_j + f_{j+1})g_{j+1}^n \hat{Q}_{j+1}^n \right] = \Delta(\hat{\rho}v)^n
\]

In the above equations, the subscripts $i$ and $j$ represent grid point indices in the $\xi$ and $\eta$ directions. For notational convenience, terms without an explicitly written $i$ or $j$ subscript are understood to be at $i$ or $j$.

On the left hand side, $f$ is the coefficient of $\partial/\partial \xi$ (or $\partial/\partial \eta$, depending on the sweep) in the $\partial \hat{E}_i/\partial \hat{Q}$ (or $\partial \hat{F}_i/\partial \hat{Q}$) Jacobian coefficient matrix. Similarly, $g$ is the term in the parentheses following $\partial/\partial \xi$ (or $\partial/\partial \eta$) in the $\partial \hat{E}_i/\partial \hat{Q}$ (or $\partial \hat{F}_i/\partial \hat{Q}$) Jacobian coefficient matrix.

The vector of dependent variables is

\[
\hat{Q} = \begin{bmatrix} \rho & \rho u & \rho v & E_T \end{bmatrix}^T
\]

The appropriate elements of the inviscid flux vectors are given by

\[
\hat{E}_3 = \frac{1}{f} \left[ \rho u \xi_x + (\rho v^2 + p)\xi_y + \rho v \xi_z \right]
\]

\[
\hat{F}_3 = \frac{1}{f} \left[ \rho u \eta_x + (\rho v^2 + p)\eta_y + \rho v \eta_z \right]
\]

The appropriate elements of the non-cross derivative viscous flux vectors are

\[
(\hat{E}_i)_3 = \frac{1}{f} \left[ 2\mu \xi_x \eta_x + \lambda \xi_y (\xi_x u_x + \xi_y v_x) + \mu \xi_x (\xi_x u_x + \xi_x v_x) \right]
\]

\[
(\hat{F}_i)_3 = \frac{1}{f} \left[ 2\mu \eta_y \eta_y + \lambda \eta_x (\eta_x u_x + \eta_x v_x) + \mu \eta_x (\eta_x u_x + \eta_x v_x) \right]
\]

And the appropriate elements of the cross derivative viscous flux vectors are

\[
(\hat{E}_x)_3 = \frac{1}{f} \left[ 2\mu \xi_x v_x + \lambda \xi_y (\xi_x u_x + \eta_x v_x) + \mu \xi_x (\xi_x u_x + \eta_x v_x) \right]
\]

\[
(\hat{F}_x)_3 = \frac{1}{f} \left[ 2\mu \eta_x v_x + \lambda \eta_y (\xi_x u_x + \eta_x v_x) + \mu \eta_x (\xi_x u_x + \eta_x v_x) \right]
\]

The elements of the Jacobian coefficient matrix $\partial \hat{E}/\partial \hat{Q}$ for the inviscid terms in the $y$-momentum equation are

\[
\frac{\partial \hat{E}_3}{\partial \hat{Q}} = \begin{bmatrix} \frac{\partial p}{\partial \rho} \xi_y - f_1 & v_x + \frac{\partial p}{\partial (p\rho)} \xi_y & \xi_x + f_1 + v_x \xi_y + \frac{\partial p}{\partial (p\rho)} \xi_y & \frac{\partial p}{\partial E_T} \xi_y \end{bmatrix}
\]

where $f_1 = u \xi_x + v \xi_y$. 

Proteus 2-D Programmer's Reference 4.0 Proteus Subprograms: COEFY 167
The elements of the Jacobian coefficient matrix $\partial \hat{E}_{\nu_1}/\partial \hat{Q}$ for the viscous terms are

$$\frac{\partial (\hat{E}_{\nu_1})_3}{\partial \hat{Q}} = \frac{1}{Re_r} \left[ \begin{array}{ccc} \frac{\partial (\hat{E}_{\nu_1})_1}{\partial \hat{Q}} & \alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{1}{\rho} \right) & \alpha_{yy} \frac{\partial}{\partial \xi} \left( \frac{1}{\rho} \right) & 0 \end{array} \right]$$

where

$$\left( \frac{\partial (\hat{E}_{\nu_1})_1}{\partial \hat{Q}} \right) = - \alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{u}{\rho} \right) - \alpha_{yy} \frac{\partial}{\partial \xi} \left( \frac{v}{\rho} \right)$$

$$\alpha_{xy} = (\mu + \lambda) \xi_x \xi_y$$

$$\alpha_{yy} = \mu \xi_x^2 + (2\mu + \lambda) \xi_y^2$$

The Jacobian coefficient matrices $\partial \hat{E}_3/\partial \hat{Q}$ and $\partial (\hat{E}_{\nu_1})_3/\partial \hat{Q}$ have the same form as $\partial \hat{E}_3/\partial \hat{Q}$ and $\partial (\hat{E}_{\nu_1})_3/\partial \hat{Q}$, but with $\xi$ replaced by $\eta$.

As an example of how these equations are translated into Fortran, consider the $\Delta (\rho u_1)$ term on the left hand side for the first sweep. This is the second element of $\hat{Q}$, so using the second element in $\partial \hat{E}_3/\partial \hat{Q}$, we get for the inviscid term

$$A(IV,I,NYM,NRU) = \frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left[ \left( \nu \xi_x \right)_{i-1,j} + \left( \frac{\partial p}{\partial (\rho u)} \xi_y \right)_{i-1,j} \right]$$

$$B(IV,I,NYM,NRU) = 0$$

$$C(IV,I,NYM,NRU) = \frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2\Delta \xi} \left[ \left( \nu \xi_x \right)_{i+1,j} + \left( \frac{\partial p}{\partial (\rho u)} \xi_y \right)_{i+1,j} \right]$$

For the viscous terms on the left hand side, we use the second element in $\partial (\hat{E}_{\nu_1})_3/\partial \hat{Q}$, which is

$$\frac{1}{Re_r} \alpha_{xy} \frac{\partial}{\partial \xi} \left( \frac{1}{\rho} \right)$$

Thus $f = \alpha_{xy}/Re_r$, and $g = 1/\rho$. To add the viscous contribution to this part of the $A$ coefficient submatrix, we therefore set

$$A(IV,I,NYM,NRU) = A(IV,I,NYM,NRU) - \frac{\theta_1(\Delta \tau)_{i,j}}{(1 + \theta_2)2(\Delta \xi)^2 Re_r} \left[ (\alpha_{xy})_{i-1,j} + \frac{\partial}{\partial \xi} \right] \frac{\left( \frac{1}{\rho} \right)}{i-1,j}$$

Similar equations may be written for the $B$ and $C$ coefficient submatrices.

The equations for axisymmetric flow are developed in Appendix B of Volume 1. The axisymmetric $r$-momentum equation for the two ADI sweeps is given by

---

21 These equations are written for the general case with swirl. For a non-swirl case, the swirl momentum $\rho w$ would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.
Sweep 1 (ξ direction)

\[
\Delta (\vec{\rho} \nu)^* + \frac{\theta_1 \Delta t}{(1 + \theta_2)2 \Delta \xi} \frac{1}{r} \left[ \left( r \frac{\partial \hat{E}_3}{\partial \hat{Q}} \right) \Delta \hat{Q}_j^* - \left( r \frac{\partial \hat{E}_3}{\partial \hat{Q}} \right) \Delta \hat{Q}_{j-1}^* \right] \\
- \frac{\theta_1 \Delta t}{(1 + \theta_2)2 \Delta \xi} \frac{1}{r} \left[ (r_{j-1} f_{j-1} + r_{j} f_{j}) \hat{b}_{j}^* \Delta \hat{Q}_j^* - (r_{j-1} f_{j-1} + 2r_{j} f_{j} + r_{j+1} f_{j+1}) \hat{b}_{j+1}^* \Delta \hat{Q}_{j+1}^* \right] \\
+ \frac{\theta_1 \Delta t}{1 + \theta_2} \left[ \frac{\partial \hat{H}_3}{\partial \hat{Q}} - \frac{\partial \hat{H}_3}{\partial \hat{Q}} \right] \Delta \hat{Q}_j^* = \frac{\Delta t}{1 + \theta_2} \left[ \delta_{p}[r \hat{E}_3] + \delta_{p}[r \hat{E}_3] + \hat{H}_3 \right]^{*}
\]

\[
+ \frac{\Delta t}{1 + \theta_2} \frac{1}{r} \left[ \delta_{p}[r \hat{E}_3] + \delta_{p}[r \hat{E}_3] \right]^{*} + (1 + \theta_2) \Delta t \frac{1}{r} \left[ \delta_{p}[r \hat{E}_3] + \delta_{p}[r \hat{E}_3] \right]^{*}
\]

\[
- \frac{\theta_1 \Delta t}{1 + \theta_2} \frac{1}{r} \left[ \delta_{p}[r \hat{E}_3] + \delta_{p}[r \hat{E}_3] \right]^{*} - \frac{\theta_2}{1 + \theta_2} \Delta (\vec{\rho} \nu)^{-1}
\]

Sweep 2 (η direction)

\[
\Delta (\vec{\rho} \nu)^* + \frac{\theta_1 \Delta t}{(1 + \theta_2)2 \Delta \eta} \frac{1}{r} \left[ \left( r \frac{\partial \hat{E}_3}{\partial \hat{Q}} \right) \Delta \hat{Q}_j^* - \left( r \frac{\partial \hat{E}_3}{\partial \hat{Q}} \right) \Delta \hat{Q}_{j-1}^* \right] \\
- \frac{\theta_1 \Delta t}{(1 + \theta_2)2 \Delta \eta} \frac{1}{r} \left[ (r_{j-1} f_{j-1} + r_{j} f_{j}) \hat{b}_{j}^* \Delta \hat{Q}_j^* - (r_{j-1} f_{j-1} + 2r_{j} f_{j} + r_{j+1} f_{j+1}) \hat{b}_{j+1}^* \Delta \hat{Q}_{j+1}^* \right] \\
+ \frac{\theta_1 \Delta t}{1 + \theta_2} \left[ \frac{\partial \hat{H}_3}{\partial \hat{Q}} - \frac{\partial \hat{H}_3}{\partial \hat{Q}} \right] \Delta \hat{Q}_j^* = \frac{\Delta t}{1 + \theta_2} \left[ \delta_{p}[r \hat{E}_3] + \delta_{p}[r \hat{E}_3] + \hat{H}_3 \right]^{*}
\]

where now

\[
\hat{Q} = \frac{1}{J} [\rho \ \rho u \ \rho v \ \rho w \ \ E_T]^{T}
\]

\[
\hat{E}_3 = \frac{1}{J} [\rho u \xi_x + (\rho v^2 + p) \xi_x + \rho v \xi_x]
\]

\[
\hat{F}_3 = \frac{1}{J} [\rho u \eta_x + (\rho v^2 + p) \eta_x + \rho v \eta_x]
\]

\[
(\hat{E}_\nu)_3 = \frac{1}{J} \frac{1}{Re_x} \left\{ 2 \mu \xi_x \nu_x + \lambda \xi_x \nu_x + \frac{1}{r} \xi_x (r \nu_x) \xi_x + \mu \xi_x (\xi_x \nu_x) + \xi_x \nu_x \right\}
\]

\[
(\hat{F}_\nu)_3 = \frac{1}{J} \frac{1}{Re_x} \left\{ 2 \mu \eta_x \nu_x + \lambda \eta_x \nu_x + \frac{1}{r} \eta_x (r \nu_x) \eta_x + \mu \eta_x (\eta_x \nu_x) + \eta_x \nu_x \right\}
\]

\[
(\hat{E}_\nu)_2 = \frac{1}{J} \frac{1}{Re_x} \left\{ 2 \mu \xi_x \eta_x + \lambda \xi_x \eta_x + \frac{1}{r} \eta_x (r \eta_x) \xi_x + \mu \xi_x (\xi_x \eta_x) + \xi_x \eta_x \right\}
\]

\[
(\hat{F}_\nu)_2 = \frac{1}{J} \frac{1}{Re_x} \left\{ 2 \mu \eta_x \xi_x + \lambda \eta_x \xi_x + \frac{1}{r} \eta_x (r \xi_x) \eta_x + \mu \eta_x (\eta_x \xi_x) + \eta_x \xi_x \right\}
\]

\[
\hat{H}_3 = \frac{1}{J} (- p - \rho \omega^2)
\]
\[ \hat{H}_{0} = \frac{1}{J} \frac{1}{Re} \left\{ -2\mu \frac{\nu}{r} - \lambda (\xi \nu^{2} + \eta \eta^{2}) + \frac{\lambda}{r} \left[ \xi (\nu^{2})_{z} + \eta (\nu^{2})_{\eta} \right] \right\} \]

The \( \hat{H} \) and \( \hat{H}_{0} \) terms, which do not appear in the 2-D planar form of the equations, result from the non-conservative form of the axisymmetric equations.

The elements of the Jacobian coefficient matrix \( \frac{\partial \hat{E}_{3}}{\partial \hat{Q}} \) for the inviscid terms in the axisymmetric form of the energy equation are

\[
\frac{\partial \hat{E}_{3}}{\partial \hat{Q}} = \left[ \frac{\partial p}{\partial \rho} \xi_{r} - \nu f_{1} \quad \nu \xi_{x} + \frac{\partial p}{\partial (\rho u)} \xi_{r} \quad \xi_{t} + f_{1} + \nu \xi_{r} + \frac{\partial p}{\partial (\rho v)} \xi_{r} \quad \frac{\partial p}{\partial (\rho w)} \xi_{r} \quad \frac{\partial p}{\partial E_{T}} \xi_{r} \right]
\]

where \( f_{1} = u \xi_{x} + v \xi_{r} \).

The elements of the Jacobian coefficient matrix \( \frac{\partial \hat{E}_{3}}{\partial \hat{Q}} \) for the viscous terms are

\[
\frac{\partial \hat{E}_{3}}{\partial \hat{Q}} = \frac{1}{Re} \left[ \left( \frac{\partial \hat{E}_{2}}{\partial \hat{Q}} \right)_{31} \right.
\]

where

\[
\left( \frac{\partial \hat{E}_{2}}{\partial \hat{Q}} \right)_{31} = -a_{x} \frac{\partial \xi}{\partial \xi} \left( \frac{1}{\rho} \right) - a_{r} \frac{\partial \xi}{\partial \xi} \left( \frac{1}{\rho} \right) - a_{r} \frac{\partial \xi}{\partial \xi} \left( \frac{1}{\rho} \right)
\]

As in 2-D planar flow, the Jacobian coefficient matrices \( \frac{\partial \hat{F}_{3}}{\partial \hat{Q}} \) and \( \frac{\partial \hat{E}_{3}}{\partial \hat{Q}} \) have the same form as \( \frac{\partial \hat{E}_{3}}{\partial \hat{Q}} \) and \( \frac{\partial \hat{E}_{3}}{\partial \hat{Q}} \), but with \( \xi \) replaced by \( \eta \).

The elements of the Jacobian coefficient matrix \( \frac{\partial \hat{H}}{\partial \hat{Q}} \) are

\[
\frac{\partial \hat{H}}{\partial \hat{Q}} = \left[ - \frac{\partial p}{\rho} + w^{2} - \frac{\partial p}{\partial (\rho u)} \quad - \frac{\partial p}{\partial (\rho v)} \quad - \frac{\partial p}{\partial (\rho w)} - 2w \quad \frac{\partial p}{\partial E_{T}} \right]
\]

The elements of the Jacobian coefficient matrix \( \frac{\partial \hat{H}_{V}}{\partial \hat{Q}} \) are

\[
\frac{\partial \hat{H}_{V}}{\partial \hat{Q}} = \frac{1}{Re} \left[ \left( \frac{\partial \hat{H}_{V}}{\partial \hat{Q}} \right)_{31} \quad \left( \frac{\partial \hat{H}_{V}}{\partial \hat{Q}} \right)_{32} \quad \left( \frac{\partial \hat{H}_{V}}{\partial \hat{Q}} \right)_{33} \right]
\]

where
Note that, except for the additional $\dot{H}$ and $\dot{H}_r$ terms, the equations for 2-D planar and axisymmetric flow are very similar. In the axisymmetric equations there are some additional terms involving the radius $r$ in the viscous flux vectors, with corresponding terms in the Jacobian coefficient matrices. The radius $r$ appears as an additional coefficient in front of the flux vectors $\dot{E}_i$, $\dot{E}_{vi}$, etc., and in front of the Jacobian coefficient matrices $\partial \dot{E}_i/\partial \dot{Q}$, $\partial (\dot{E}_{vi})_i/\partial \dot{Q}$, etc. In addition, $1/r$ appears in front of every term in the equation except the $\Delta(\dot{\rho}v)$ terms. In Proteus, the Fortran variables are defined in such a way that, for many terms, the same coding can be used for both 2-D planar and axisymmetric flow. Unfortunately, this may make some of the coding a little confusing. It is hoped that this detailed description, when compared with the source listing, will help make things clear.

In COEFY, the coefficients of the left hand side, or implicit, terms are defined first. Note that the implicit terms for the second ADI sweep have exactly the same form as for the first sweep, but with $\zeta$ replaced by $\eta$. By defining DEL, METX, METY, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for both sweeps. The variable RAX is equal to 1 for 2-D planar flow, and the radius $r$ for axisymmetric flow. This adds the $r$ in front of the Jacobian coefficient matrices for axisymmetric flow, but has no effect for 2-D planar flow. The $1/r$ coefficient in front of each term will be added later. In this section of code, the coefficient of $\Delta(\dot{\rho}v)$ (part of $B(IV,I,NYM,NRV)$) is set equal to $r$, not 1 as it should be. This will be corrected later.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms. These formulas are presented in Section 5.0 of Volume 1. For axisymmetric flow, the Fortran variable JI, which is normally defined as $1/r$, is temporarily redefined as $r/J$ before the COEF routines are called. This automatically accounts for the $r$ coefficient in front of all the flux vectors in the source term. The $1/r$ coefficient in front of each term will be added later. This definition of JI adds an $r$ in front of the $\Delta(\dot{\rho}v)$ term that should not be there. This will also be corrected later.

The coding for the source term for the second sweep comes next. The definition of JI also adds an $r$ in front of the $\Delta(\dot{\rho}v)$ term that should not be there.

And finally, for axisymmetric flow, the entire equation is divided by the local radius $r$. This adds the $1/r$ coefficient where it should be added, and removes the $r$ in front of the $\Delta(\dot{\rho}v)$ terms.

Remarks

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables $A$, $B$, $C$, and $S$ may be confusing. The first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. For sections of the code that apply to both sweeps (i.e., the implicit terms and the division by $r$ at the end), the first two subscripts are written as $(IV,I)$. For sections of the code that apply only
to the first sweep, the first two subscripts are written as (I2, I1). For sections that apply only to the second sweep, they are written as (I1, I2). The third subscript on A, B, C, and S corresponds to the equation. And, for A, B, and C, the fourth subscript corresponds to the dependent variable for which A, B, or C is a coefficient.

3. The coding of the extra coefficients and source terms in the axisymmetric form of the equations is separate from the rest of the coding, and is bypassed if the flow is not axisymmetric. Similarly, the coding of coefficients and source terms involving the swirl velocity is separate from the rest of the coding, and is bypassed if there is no swirl.

4. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous terms.

5. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous terms containing derivatives in the specified direction.
**Subroutine COEFZ (A,B,C,S,METX,METY,METT,NVD,NPTSD)**

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td></td>
<td>Compute coefficients and source term for the swirl momentum equation.</td>
</tr>
</tbody>
</table>

**Input**

- **DEL**
  - Computational grid spacing in sweep direction.
- **DTAU**
  - Time step $\Delta t$.
- **DXI, DETA**
  - Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **ETAX, ETAY, ETAT**
  - Metric coefficients $\eta_x, \eta_y$ (or $\eta_r$ if axisymmetric), and $\eta_t$.
- **IBASE, ISTEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **IEULER**
  - Flag for Euler calculation.
- **IHSTAG**
  - Flag for constant stagnation enthalpy option.
- **ISWEEP**
  - Current ADI sweep number.
- **ITHIN**
  - Flags for thin-layer option.
- **IV**
  - Index in the "vectorized" direction, $i$.
- **I1, I2**
  - Grid indices $i$ and $j$, in the $\xi$ and $\eta$ directions.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation times the radius, $rJ^{-1}$.
- **METX, METY, METT**
  - Derivatives of sweep direction computational coordinate with respect to $x, r$, and $t$.
- **MU**
  - Effective coefficient of viscosity $\mu$ at time level $n$.
- **NEQ**
  - Number of coupled equations being solved, $N_{eq}$.
- **NPTS**
  - Number of grid points in the sweep direction, $N$.
- **NR, NRU, NRV, NRW, NET**
  - Array indices associated with the dependent variables $p, \rho u, \rho v, \rho w$, and $E_r$.
- **NVD, NPTSD**
  - Leading two dimensions for the arrays $A, B, C, S, METX, METY$, and METT.
- **NZM**
  - Array index associated with the swirl momentum equation.
- **RAX**
  - The local radius $r$.
- **RER**
  - Reference Reynolds number $Re_r$.
- **RHO, U, V, W**
  - Static density $\rho$, and velocities $u, v$, and $w$, at time level $n$.
- **RHOL, WL**
  - Static density $\rho$ and velocity $w$ from previous ADI sweep.
- **THZ**
  - Parameters $\theta_1, \theta_2$, and $\theta_3$ determining type of time differencing for the swirl momentum equation.
- **XIX, XIY, XIT**
  - Metric coefficients $\xi_x, \xi_y$ (or $\xi_r$ if axisymmetric), and $\xi_t$.
- **Y**
  - Radial coordinate $r$. 
Output

A, B, C  Coefficient submatrices A, B, and C at interior points (row NZM only).

S  Source term subvector S at interior points (element NZM only).

Description

Subroutine COEFZ computes the coefficients and source term for the swirl momentum equation, which is only valid in axisymmetric flow. The equations for axisymmetric flow are developed in Appendix B of Volume I. The swirl momentum equation for the two ADI sweeps is given by

Sweep 1 (ξ direction)

\[
\Delta(\rho\omega)^i + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2(\Delta \xi)} \frac{1}{r} \left[ \left( \frac{\partial \hat{E}_4}{\partial Q} \right)^n_{j+1} - \left( \frac{\partial \hat{E}_4}{\partial Q} \right)^n_{j-1} \right] \\
- \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2(\Delta \xi)} \frac{1}{r} \left[ (r_{j-1}f_{j-1} + r_j f_j)g_{j-1} \Delta Q_{j-1} - (r_{j-1}f_{j-1} + 2r_j f_j + r_{j+1} f_{j+1})g_{j+1} \Delta Q_{j+1} + (r_{j+1} f_{j+1})g_{j+1} \Delta Q_{j+1} \right] \\
+ \frac{\theta_1 \Delta \tau}{1 + \theta_2} \frac{1}{r} \left[ \frac{\partial H_4}{\partial Q} - \frac{\partial (H_{PV})}{\partial Q} \right] \Delta Q_{j+1}^- = -\frac{\Delta \tau}{1 + \theta_2} \frac{1}{r} \left[ \delta_j \left( \frac{\partial \hat{E}_4}{\partial P} \right) + \delta_j \left( \frac{\partial \hat{F}_4}{\partial P} \right) \right] + \Delta \frac{\Delta \tau}{1 + \theta_2} \frac{1}{r} \left[ \delta_j \left( \frac{\partial \hat{E}_4}{\partial P} \right) + \delta_j \left( \frac{\partial \hat{F}_4}{\partial P} \right) \right]^{-1} + \frac{\theta_2 \Delta \tau}{1 + \theta_2} \Delta(\rho\omega)^i_{j+1}
\]

Sweep 2 (η direction)

\[
\Delta(\rho\omega)^j + \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2(\Delta \eta)} \frac{1}{r} \left[ \left( \frac{\partial \hat{E}_4}{\partial Q} \right)^n_{j+1} - \left( \frac{\partial \hat{E}_4}{\partial Q} \right)^n_{j-1} \right] \\
- \frac{\theta_1 \Delta \tau}{(1 + \theta_2)2(\Delta \eta)} \frac{1}{r} \left[ (r_{j-1}f_{j-1} + r_j f_j)g_{j-1} \Delta Q_{j-1} - (r_{j-1}f_{j-1} + 2r_j f_j + r_{j+1} f_{j+1})g_{j+1} \Delta Q_{j+1} + (r_{j+1} f_{j+1})g_{j+1} \Delta Q_{j+1} \right] \\
+ \frac{\theta_1 \Delta \tau}{1 + \theta_2} \frac{1}{r} \left[ \frac{\partial H_4}{\partial Q} - \frac{\partial (H_{PV})}{\partial Q} \right] \Delta Q_{j+1}^- = \frac{\theta_1 \Delta \tau}{1 + \theta_2} \frac{1}{r} \left[ \frac{\partial H_4}{\partial Q} - \frac{\partial (H_{PV})}{\partial Q} \right] \Delta Q_{j+1}^- + \frac{\theta_2 \Delta \tau}{1 + \theta_2} \Delta(\rho\omega)^i_{j+1}
\]

In the above equations, the subscripts i and j represent grid point indices in the ξ and η directions. For notational convenience, terms without an explicitly written i or j subscript are understood to be at i or j.

On the left hand side, f is the coefficient of \( \partial / \partial \xi \) (or \( \partial / \partial \eta \), depending on the sweep) in the \( \partial \hat{E}_4 / \partial Q \) (or \( \partial \hat{F}_4 / \partial Q \)) Jacobian coefficient matrix. Similarly, g is the term in the parentheses following \( \partial / \partial \xi \) (or \( \partial / \partial \eta \)) in the \( \partial \hat{E}_4 / \partial Q \) (or \( \partial \hat{F}_4 / \partial Q \)) Jacobian coefficient matrix.

The vector of dependent variables is

\[
\hat{Q} = \left[ \rho \quad \rho u \quad \rho v \quad \rho w \quad E_T \right]^T
\]

The appropriate elements of the inviscid flux vectors are given by

---

24 These equations are written assuming the energy equation is being solved. For a constant stagnation enthalpy case, the total energy \( E_T \) would not appear as a dependent variable, and the Jacobian coefficient matrices would have only four elements.
The appropriate elements of the non-cross derivative viscous flux vectors are

$$
\hat{E}_V = \frac{1}{\mathcal{F}} \left[ \rho v w_x + \rho v w_r + \rho \xi w_r \right] \\
\hat{F}_V = \frac{1}{\mathcal{F}} \left[ \rho w w_x + \rho w w_r + \rho \eta w_r \right]
$$

And the appropriate elements of the cross derivative viscous flux vectors are

$$
\hat{E}_V = \frac{1}{\mathcal{F}} \left[ \mu \xi \eta w_x + \mu \xi \eta w_r - \mu \xi \frac{w}{r} \right] \\
\hat{F}_V = \frac{1}{\mathcal{F}} \left[ \mu \eta \xi w_x + \mu \eta \xi w_r - \mu \eta \frac{w}{r} \right]
$$

The extra terms resulting from the non-conservative form of the axisymmetric equations are

$$
\hat{H}_V = \frac{1}{\mathcal{F}} \rho w w
$$

The elements of the Jacobian coefficient matrix $\partial \hat{E}/\partial \hat{Q}$ for the inviscid terms in the $r$-momentum equation are

$$
\frac{\partial \hat{E}_V}{\partial \hat{Q}} = \frac{1}{\mathcal{F}} \left[ - \mathcal{F} f_i \begin{bmatrix} w \xi_x & w \xi_r & \xi + f_i & 0 \end{bmatrix} \right]
$$

where $f_i = u \xi_x + v \xi_r$.

The elements of the Jacobian coefficient matrix $\partial \hat{E}_V/\partial \hat{Q}$ for the viscous terms are

$$
\frac{\partial \hat{E}_V}{\partial \hat{Q}} = \frac{1}{\mathcal{F}} \left[ \begin{bmatrix} \mathcal{F} f_i \end{bmatrix} \begin{bmatrix} \frac{\partial \hat{E}_V}{\partial \hat{Q}} \end{bmatrix} \begin{bmatrix} \mathcal{F} f_i \end{bmatrix} \right]
$$

where

$$
\frac{\partial \hat{E}_V}{\partial \hat{Q}} = - a_{zz} \frac{\partial}{\partial \xi} \left( \frac{w}{\rho} \right)
$$

$$
a_{zz} = \mu \xi_x^2 + \mu \xi_r^2
$$

The Jacobian coefficient matrices $\partial \hat{F}_V/\partial \hat{Q}$ and $\partial (\hat{F}_V)/\partial \hat{Q}$ have the same form as $\partial \hat{E}_V/\partial \hat{Q}$ and $\partial (\hat{E}_V)/\partial \hat{Q}$, but with $\xi$ replaced by $\eta$. 
The elements of the Jacobian coefficient matrix $\frac{\partial \mathbf{H}}{\partial \mathbf{Q}}$ are

\[
\frac{\partial \mathbf{H}_4}{\partial \mathbf{Q}} = \begin{bmatrix}
-\nu w & 0 & w
\end{bmatrix}
\]

The elements of the Jacobian coefficient matrix $\frac{\partial \mathbf{H}_v}{\partial \mathbf{Q}}$ are

\[
\frac{\partial (\mathbf{H}_v)_4}{\partial \mathbf{Q}} = \frac{1}{Re_r} \begin{bmatrix}
(\frac{\partial \mathbf{H}_v}{\partial \mathbf{Q}})_{41} & 0 & (\frac{\partial \mathbf{H}_v}{\partial \mathbf{Q}})_{44}
\end{bmatrix}
\]

where

\[
(\frac{\partial \mathbf{H}_v}{\partial \mathbf{Q}})_{41} = -\mu_r \frac{\partial}{\partial \xi} \left( \frac{w}{\rho} \right) + \mu_r \frac{w}{r} - \mu_r \frac{\partial}{\partial \eta} \left( \frac{w}{\rho} \right)
\]

\[
(\frac{\partial \mathbf{H}_v}{\partial \mathbf{Q}})_{44} = \mu_r \frac{\partial}{\partial \xi} \left( \frac{1}{\rho} \right) - \mu_r \frac{1}{r} + \mu_r \frac{\partial}{\partial \eta} \left( \frac{1}{\rho} \right)
\]

As an example of how these equations are translated into Fortran, consider the $A(p_v/J)$ term on the left hand side for the first sweep. This is the third element of $\mathbf{Q}$, so using the third element in $\frac{\partial \mathbf{F}_4}{\partial \mathbf{Q}}$, and including the contribution from the third element of $\frac{\partial \mathbf{H}_v}{\partial \mathbf{Q}}$, we get for the inviscid term

\[
A(IV,I,NZM,NRV) = -\frac{\theta_1(\Delta r)_{i,j}}{(1 + \theta_2)2\Delta \xi} \frac{1}{r_{i,j}} [(r\xi)_i - 1, j]
\]

\[
B(IV,I,NZM,NRV) = \frac{\theta_1(\Delta r)_{i,j}}{1 + \theta_2} \frac{1}{r_{i,j}} w_{i,j}
\]

\[
C(IV,I,NZM,NRV) = \frac{\theta_1(\Delta r)_{i,j}}{(1 + \theta_2)2\Delta \xi} \frac{1}{r_{i,j}} [(r\xi)_i + 1, j]
\]

For the $\Delta(p_v/J)$ term, there are no viscous terms on the left hand side.

In COEFZ, the coefficients of the left hand side, or implicit, terms are defined first. Note that the implicit terms for the second sweep have exactly the same form as for the first sweep, but with $\xi$ replaced by $\eta$. By defining DEL, METX, METY, and METT as the grid spacing and metric coefficients in the sweep direction, the same coding can be used for both sweeps. Since COEFZ is only used in axisymmetric flow, the variable RAX is equal to the radius $r$. This adds the $r$ in front of the Jacobian coefficient matrices. The $1/r$ coefficient in front of each term will be added later. In this section of code, the coefficient of $\Delta(\rho \omega)$ (part of $B(IV,I,NZM,NRW)$) is set equal to $r$, not $1$ as it should be. This will be corrected later.

The source term, or right hand side, for the first sweep is defined next. The difference formulas used to compute the source term are the same as those used for the implicit terms. These formulas are presented in Section 5.0 of Volume 1. For axisymmetric flow, the Fortran variable JJ, which is normally defined as $1/J$, is temporarily redefined as $r/J$ before the COEF routines are called. This automatically accounts for the $r$ coefficient in front of all the flux vectors in the source term. The $1/r$ coefficient in front of each term will be added later. This definition of JJ adds an $r$ in front of the $\Delta(\rho \omega)^{-1}$ term that should not be there. This will also be corrected later.
The coding for the source term for the second sweep comes next. The definition of \( J_l \) also adds an \( r \) in front of the \( \Delta (\rho \omega)^* \) term that should not be there.

And finally, the entire equation is divided by the local radius \( r \). This adds the \( 1/r \) coefficient where it should be added, and removes the \( r \) in front of the \( \Delta (\rho \omega) \) terms.

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
2. The subscripts on the Fortran variables \( A, B, C, \) and \( S \) may be confusing. The first subscript is the index in the non-sweep (i.e., "vectorized") direction, and the second subscript is the index in the sweep direction. For sections of the code that apply to both sweeps (i.e., the implicit terms and the division by \( r \) at the end), the first two subscripts are written as \((IV,I)\). For sections of the code that apply only to the first sweep, the first two subscripts are written as \((I2,I)\). For sections that apply only to the second sweep, they are written as \((I1,I2)\). The third subscript on \( A, B, C, \) and \( S \) corresponds to the equation. And, for \( A, B, \) and \( C \), the fourth subscript corresponds to the dependent variable for which \( A, B, \) or \( C \) is a coefficient.
3. The coding of the extra coefficients and source terms in the axisymmetric form of the equations is separate from the rest of the coding, and is bypassed if the flow is not axisymmetric. Similarly, the coding of coefficients and source terms involving the swirl velocity is separate from the rest of the coding, and is bypassed if there is no swirl.
4. The Euler option is implemented simply by skipping the calculation of the coefficients and source terms for the viscous terms.
5. The thin-layer option is implemented by skipping the calculation of the coefficients and source terms for the viscous terms containing derivatives in the specified direction.
### Subroutine CONV

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>ISAMAX</td>
<td>Test computed flow field for convergence.</td>
</tr>
</tbody>
</table>

#### Input

**CHGMAX**

Maximum change in absolute value of the dependent variables from time level \( n - 1 \) to \( n \) (or over the previous NITAVG – 1 time steps if ICTEST = 2), \( \Delta Q_{\text{max}} \).

**DUMMY**

A two-dimensional scratch array.

**EPS**

Convergence level to be reached, \( \varepsilon \).

**GAMR**

Reference ratio of specific heats, \( \gamma_r \).

**IAV2E, IAV4E**

Flags for second- and fourth-order explicit implicit artificial viscosity.

**ICTEST**

Flag for convergence criteria to be used.

**IHSTAG**

Flag for constant stagnation enthalpy option.

**ISWIRL**

Flag for swirl in axisymmetric flow.

**IT**

Current time step number \( n \).

**NEQ**

Number of coupled equations being solved, \( N_{eq} \).

**NITAVG**

Number of time steps in moving average convergence test.

**NOUT**

Unit number for standard output.

**NR, NRU, NRV, NRW, NET**

Array indices associated with the dependent variables \( \rho, \rho u, \rho v, \rho w, \) and \( E_r \).

**NTOTP**

Dimensioning parameter specifying the storage required for a full two-dimensional array (i.e., \( N_{1P} \times N_{2P} \)).

**N1, N2**

Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions.

**RESAVG**

The average absolute value of the residual at time level \( n \), \( R_{eq} \).

**RESL2**

The \( L_2 \) norm of the residual at time level \( n \), \( R_{l2} \).

**RESMAX**

The maximum absolute value of the residual at time level \( n \), \( R_{max} \).

**RGAS**

Gas constant \( R \).

**RHO, U, V, W, ET**

Static density \( \rho \), velocities \( u, v, \) and \( w \), and total energy \( E_r \) at time level \( n + 1 \).

**RHOL, UL, VL, WL, ETL**

Static density \( \rho \), velocities \( u, v, \) and \( w \), and total energy \( E_r \) at time level \( n \).

#### Output

**CHGAVG**

Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps, \( \Delta Q_{av} \).

**CHGMAX**

Maximum change in absolute value of the dependent variables from time level \( n \) to \( n + 1 \) (or over the previous NITAVG time steps if ICTEST = 2), \( \Delta Q_{\text{max}} \).
Description

Subroutine CONV checks the computed flow field for convergence. Convergence may be based on: (1) the absolute value of the maximum change in the dependent variables over the previous time step; (2) the absolute value of the maximum change in the dependent variables, averaged over the last NITAVG time steps; (3) the $L_2$ norm of the residual for each equation; (4) the average residual for each equation; or (5) the maximum residual for each equation. These parameters are defined in Section 4.1.6 of Volume 2.

The convergence criteria to be used and the level to be reached are set by the input parameters ICTEST and EPS. Each dependent variable or equation is checked separately, and convergence is declared when the specified level is reached for all of the variables or equations. The same criteria is used for each one, but different levels may be specified.

Subroutine CONV first computes $\Delta Q_{max}$, the absolute value of the maximum change in each dependent variable over all the grid points for the most recent time step. These values are stored in CHGMAX(IVAR,1), where IVAR varies from 1 to NEQ, the number of dependent variables. If ICTEST = 2 (the so-called "moving average" convergence test), CHGMAX(IVAR,2) contains the maximum change for the previous time step, etc.

Then, depending on the value of ICTEST, the chosen convergence criteria is compared with the level to be reached for each dependent variable or equation, and a flag is set if the calculation is converged.

Remarks

1. For ICTEST = 1 or 2, the change in $E_T$ is divided by $R/((\gamma - 1) + 1/2)$. This is equivalent to dividing the dimensional value $E_T$ by

$$E_T = \frac{\rho \bar{R} T_r}{\gamma - 1} + \frac{\rho \mu_U^2}{2}$$

This makes the change in total energy the same order of magnitude as the other conservation variables.

2. For ICTEST = 1 or 2, the convergence test is based on (or includes) the change in dependent variables from time level $n$ to $n + 1$. For ICTEST = 3, 4, or 5, convergence is based on the residual at time level $n$, not $n + 1$. This is because the residuals at time level $n + 1$ are not computed until the marching step from $n + 1$ to $n + 2$ is taken.

3. For cases run with artificial viscosity, the residuals are computed and printed both with and without the artificial viscosity terms. This may provide some estimate of the overall error in the solution introduced by the artificial viscosity. Convergence is determined by the residuals with the artificial viscosity terms included.

4. The Cray search routine ISAMAX is used in computing the absolute value of the maximum change in dependent variables.

5. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the change in dependent variables for use by ISAMAX.

6. A warning message is generated if an illegal convergence criteria is specified. ICTEST is reset to 3 (convergence based on the $L_2$ norm of the residual), and the calculation will continue.
Subroutine CUBIC (IDIR,T,G,NOLD,TINT,GINT)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAK</td>
<td></td>
<td>Interpolation using Ferguson's parametric cubic.</td>
</tr>
</tbody>
</table>

**Input**

- **G**
  A two-dimensional array containing NOLD1 x NOLD2 values of the function g(t) to be interpolated.

- **IDIR**
  Direction flag; 1 if first subscript in G varies, 2 if second subscript varies.

- **I1, I2**
  Grid indices i and j, in the \( \xi \) and \( \eta \) directions.

- **NOLD**
  Number of values in direction IDIR in array G (i.e., NOLD1 or NOLD2.)

- **N1, N2**
  Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions.

- **T**
  A one-dimensional array containing NOLD values of the independent variable \( t \).

- **TINT**
  A one-dimensional array containing N1 or N2 (depending on IDIR) values of the independent variable \( t = t_{in} \) at which interpolated values \( g_{in} = g(t_{in}) \) are desired.

**Output**

- **GINT**
  A one-dimensional array containing N1 or N2 (depending on IDIR) interpolated values \( g_{in} = g(t_{in}) \).

**Description**

Subroutine CUBIC performs interpolation using Ferguson's parametric cubic polynomial (Faux and Pratt, 1979). Given the function \( g(t) \) and a value \( t_{in} \), CUBIC computes \( g_{in} = g(t_{in}) \).

The function \( g(t) \) is specified by the Fortran arrays G and T. For a general value \( t \), let

\[
 t_f = \frac{t - t_u}{t_d - t_u}
\]

where \( t_u \leq t \leq t_d \). (i.e., \( t_u \) and \( t_d \) are the two elements of the array T that bracket \( t \).)

Between \( t_u \) and \( t_d \), assume \( g \) can be described by a cubic polynomial in \( t_f \), as follows:

\[
g = a_1 + a_2 t_f + a_3 t_f^2 + a_4 t_f^3
\]

Differentiating,

\[
g' = \frac{dg}{dt_f} = a_2 + 2a_3 t_f + 3a_4 t_f^2
\]

Noting that \( t_f = 0 \) at \( t = t_u \), and 1 at \( t = t_d \), we get
\[
\begin{align*}
g_u &= a_1 \\
g'_u &= a_2 \\
g_d &= a_1 + a_2 + a_3 + a_4 \\
g'_d &= a_2 + 2a_3 + 3a_4
\end{align*}
\]

Solving for \(a_1\) through \(a_4\),

\[
\begin{align*}
a_1 &= g_u \\
a_2 &= g'_u \\
a_3 &= 3(g_d - g_u) - 2g'_u - g'_d \\
a_4 &= 2(g_u - g_d) + g'_u + g'_d
\end{align*}
\]

Plugging these into the cubic polynomial for \(f\) and rearranging,

\[
g = g_u(1 - 3t^2 + 2t^3) + g_d(3t^2 - 2t^3) + g'_u(t - 2t^2 + t^3) + g'_d(-t^2 + t^3)
\]

This is the form of the equation used to compute \(g_{\text{out}}\).

**Remarks**

1. At interior points in the array \(g\), the derivatives \(g'_u\) and \(g'_d\) are computed using a second-order central difference formula. At the end points, second-order one-sided difference formulas are used.

2. The Fortran variable \(\text{TINT}\) is actually a one-dimensional array containing \(N_1\) or \(N_2\) input values of \(t_{\text{in}}\). Similarly, \(\text{GINT}\) is a one-dimensional array containing \(N_1\) or \(N_2\) output values of \(g_{\text{out}}\).

3. The Fortran array \(G\) that specifies the input values of \(g(t)\) is actually a two-dimensional array. Within \(\text{CUBIC}\), however, only one of the subscripts varies. The input flag \(\text{IDIR}\) specifies which one.
Subroutine EQSTAT (ICALL)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BVUP</td>
<td></td>
<td>Use equation of state to compute pressure, temperature, and their derivatives with respect to the dependent variables.</td>
</tr>
<tr>
<td>EXEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAIN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input

- **CP, CV**
  - Specific heats \( c_p \) and \( c_v \).
- **HSTAG**
  - Stagnation enthalpy \( h_T \) used with constant stagnation enthalpy option.
- **IBASE, ISTEP**
  - Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.
- **ICALL**
  - 0 to get \( p \) and \( T \), 1 to get derivatives of \( p \) and \( T \) with respect to dependent variables.
- **IHSTAG**
  - Flag for constant stagnation enthalpy option.
- **NPTS**
  - Number of grid points in the sweep direction, \( N \).
- **N1, N2**
  - Number of grid points \( N_1 \) and \( N_2 \) in the \( \xi \) and \( \eta \) directions.
- **RGAS**
  - Gas constant \( R \).
- **RHO, U, V, W, ET**
  - Static density \( \rho \), velocities \( u, v, \) and \( w \), and total energy \( E_T \).

Output

- **DPDRHO, DPDRU, DPDRV, DPDRW, DPDET**
  - Derivatives \( \partial p/\partial \rho \), \( \partial p/\partial (\rho u) \), \( \partial p/\partial (\rho v) \), \( \partial p/\partial (\rho w) \), and \( \partial p/\partial E_T \).
- **DTDRHO, DTDRU, DTDRV, DTDRW, DTDET**
  - Derivatives \( \partial T/\partial \rho \), \( \partial T/\partial (\rho u) \), \( \partial T/\partial (\rho v) \), \( \partial T/\partial (\rho w) \), and \( \partial T/\partial E_T \).
- **ET**
  - Total energy (constant stagnation enthalpy option only.)
- **INEG**
  - Flag for non-positive pressure and/or temperature; 0 if positive, 1 if non-positive.
- **P, T**
  - Static pressure \( p \) and temperature \( T \).

Description

Subroutine EQSTAT computes various quantities that depend on the form of the equation of state. It actually serves a dual purpose. First, it is called from subroutine INITC and from the MAIN program, with the input parameter ICALL = 0, to compute the static pressure \( p \) and temperature \( T \) from the initial or just-computed values of the dependent variables. If the constant stagnation enthalpy option is being used it also computes a value for the total energy \( E_T \). And second, it is called from subroutines BVUP and EXEC, with ICALL = 1, to compute the derivatives of \( p \) and \( T \) with respect to the dependent variables. If the constant stagnation enthalpy option is being used it also computes a value for the total energy \( E_T \). And second, it is called from subroutines BVUP and EXEC, with ICALL = 1, to compute the derivatives of \( p \) and \( T \) with respect to the dependent variables.

The equation of state currently built into Proteus is for a perfect gas. The formulas used to compute \( p, T \), and their derivatives with respect to the dependent variables are presented in Section 4.3 of Volume 1 for two-dimensional planar flow and in Section B.2.3 of Volume 1 for axisymmetric flow.

---

25 These are needed for linearization of the governing equations. See Section 4.1 of Volume 1 for details.
Remarks

1. When used to compute $p$ and $T$ ($ICALL = 0$), this subroutine is called from outside any loops in the $\xi$ or $\eta$ directions. When used to compute $\partial p/\partial \rho$, etc., ($ICALL = 1$), it is called for each ADI sweep from inside a loop in the non-sweep direction.

2. When computing $\partial p/\partial \rho$, etc., this subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.
Subroutine EXEC

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>ADI, AVISCl, AVISC2, BCELIM, BCGEN, BVUP, COEFC, COEFE, COEFX, COEFY, COEFZ, EQSTAT, PERIOD, RESID, UPDATE</td>
<td>Manage solution of governing equations.</td>
</tr>
</tbody>
</table>

**Input**

- **DXI, DETA**: Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **ETAX, ETAY, ETAT**: Metric coefficients $\eta_x, \eta_y$ (or $\eta$, if axisymmetric), and $\eta_n$.
- **IAV2E, IAV4E, IAV2I**: Flags for second-order explicit, fourth-order explicit, and second-order implicit artificial viscosity.
- **IAXI**: Flag for axisymmetric flow.
- **IBCELM**: Flags for elimination of off-diagonal coefficient submatrices resulting from three-point boundary conditions in the $\xi$ and $\eta$ directions at either boundary; 0 if elimination is not necessary, 1 if it is.
- **ICHECK**: Convergence checking interval.
- **IHSTAG**: Flag for constant stagnation enthalpy option.
- **ISWIRL**: Flag for swirl in axisymmetric flow.
- **IT**: Current time step number $n$.
- **ITBEG**: The time level $n$ at the beginning of a run.
- **ITHIN**: Flags for thin-layer option.
- **JI**: Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.
- **KBCPER**: Flags for spatially periodic boundary conditions in the $\xi$ and $\eta$ directions; 0 for non-periodic, 1 for periodic.
- **NEQP**: Dimensioning parameter specifying maximum number of coupled equations allowed.
- **NPT1, NPT2**: $N_1$ and $N_2$ for non-periodic boundary conditions, $N_1 + 1$ and $N_2 + 1$ for spatially periodic boundary conditions in $\xi$ and $\eta$.
- **NTABC**: Dimensioning parameter specifying total storage required for the coefficient submatrices A, B, and C.
- **NTS**: Dimensioning parameter specifying total storage required for the source term subvector S.
* N1, N2
  N1P, N2P
  XIX, XIY, XIT
  Y

  Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions.

  Parameters specifying the dimension sizes in the \( \xi \) and \( \eta \) directions.

  Metric coefficients \( \xi_x, \xi_y \) (or \( \xi_r \), if axisymmetric), and \( \xi_r \).

  Radial coordinate \( r \) for axisymmetric flow.

**Output**

<table>
<thead>
<tr>
<th>DEL</th>
<th>Computational grid spacing in sweep direction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBASE, ISTEP</td>
<td>Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.</td>
</tr>
<tr>
<td>ISWEEP</td>
<td>Current ADI sweep number.</td>
</tr>
<tr>
<td>IV</td>
<td>Index in the &quot;vectorized&quot; direction, ( i ).</td>
</tr>
<tr>
<td>II, I2</td>
<td>Grid indices ( i ) and ( j ), in the ( \xi ) and ( \eta ) directions.</td>
</tr>
<tr>
<td>JI</td>
<td>The radius times the inverse Jacobian of the nonorthogonal grid transformation, ( rJ^{-1} ) (used in COEF routines for axisymmetric flow only.)</td>
</tr>
<tr>
<td>METX, METY, METT</td>
<td>Derivatives of sweep direction computational coordinate with respect to ( x, y ) (or ( r ) if axisymmetric), and ( t ).</td>
</tr>
<tr>
<td>NPTS</td>
<td>Number of grid points in the sweep direction, ( N ).</td>
</tr>
<tr>
<td>NV</td>
<td>Number of grid points in the &quot;vectorized&quot; direction, ( N_v ).</td>
</tr>
<tr>
<td>RAX</td>
<td>1 for two-dimensional planar flow, and the local radius ( r ) for axisymmetric flow.</td>
</tr>
<tr>
<td>RHO, U, V, W, ET</td>
<td>Static density ( \rho ), velocities ( u, v, ) and ( w ), and total energy ( E_T ) at time level ( n + 1 ).</td>
</tr>
<tr>
<td>RHOL, UL, VL, WL, ETL</td>
<td>Static density ( \rho ), velocities ( u, v, ) and ( w ), and total energy ( E_T ) at time level ( n ).</td>
</tr>
<tr>
<td>TL</td>
<td>Static temperature ( T ) at time level ( n ).</td>
</tr>
</tbody>
</table>

**Description**

Subroutine EXEC manages the solution of the governing equations. It is called by the MAIN program during each marching step from time level \( n \) to \( n + 1 \). The steps involved in EXEC are described below.

**Preliminary Steps**

1. If this is the first time step, temporarily set the thin-layer flags to zero.
2. Initialize the coefficient submatrices \( A, B, \) and \( C \), and the source term subvector \( S \), to zero.
3. If spatially periodic boundary conditions are being used in either direction, call PERIOD to add the appropriate extra line(s) of data.

**First ADI sweep, \( \xi \) direction**

4. Set various sweep-dependent parameters, as follows:

\[
\begin{align*}
  \text{isweep} &= 1 \\
  \text{istep} &= 1 \\
  \text{del} &= \Delta \xi \\
  \text{npts} &= N_1 \text{ or } N_1 + 1 \\
  \text{nv} &= N_2 \text{ or } N_2 + 1
\end{align*}
\]
5. For axisymmetric flow, set \( J_1 = r/J \) at all grid points.
6. Set metrics in sweep (\( \xi \)) direction at all grid points as follows:
   \[
   \begin{align*}
   \text{met}_x(i2,il) &= (\xi_x)_{il} \\
   \text{met}_y(i2,il) &= (\xi_y)_{il} \text{ or } (\xi_z)_{il} \\
   \text{met}_t(i2,il) &= (\xi_t)_{il}
   \end{align*}
   \]
7. Begin loop in non-sweep (\( \eta \)) direction over interior points (\( j = I_2 = 2 \) to \( NPT_2 - 1 \)).
8. For axisymmetric flow, set \( RAX(I1) = r_{ij} \) along the current \( \eta \)-line at all \( \xi \) grid points.
9. Call EQSTAT to get the derivatives of \( p \) and \( T \) with respect to \( \rho, \rho u \), etc., along the current \( \eta \)-line at all \( \xi \) grid points.
10. Call the COEF routines to compute the coefficients and source terms for the governing equations along the current \( \eta \)-line at all interior \( \xi \) grid points.
11. End of loop in non-sweep (\( \eta \)) direction.
12. For axisymmetric flow, reset \( J_1 = 1/J \) at all grid points.
13. For non-spatially periodic boundary conditions in the \( \xi \) direction, begin loop in non-sweep (\( \eta \)) direction over interior points (\( j = I_2 = 2 \) to \( NPT_2 - 1 \)).
14. Call EQSTAT to get the derivatives of \( p \) and \( T \) with respect to \( \rho, \rho u \), etc., along the current \( \eta \)-line at all \( \xi \) grid points.
15. Call BCGEN to compute the coefficients and source terms for the boundary condition equations at the end points (\( i = I_1 = 1 \) and \( N_1 \)) of the current \( \eta \)-line.
16. If three-point boundary conditions were used at either boundary, call BCELIM to eliminate the resulting off-diagonal coefficient submatrices.
17. End of loop in non-sweep (\( \eta \)) direction.
18. Every ICHECK time steps, call RESID to compute residuals at time level \( n \) without the artificial viscosity terms, and to update the convergence history file.
19. If artificial viscosity is being used, call AVISC1 or AVISC2 to add the appropriate terms to the coefficient submatrices and/or the source term subvectors at all interior grid points.
20. Every ICHECK time steps, if artificial viscosity is being used, call RESID to compute residuals at time level \( n \) with the artificial viscosity terms, and to update the convergence history file.
21. If spatially periodic boundary conditions are being used in the \( \xi \) direction, reset \( NPTS = N_1 \).
22. Call ADI to solve the system of difference equations.
23. Begin loop in non-sweep (\( \eta \)) direction over interior points (\( j = I_2 = 2 \) to \( NPT_2 - 1 \)).
24. Call UPDATE to compute the primitive flow variables, \( Q^* \), from the newly computed conservation variables in delta form, \( \Delta Q^* \), along the current \( \eta \)-line at all \( \xi \) grid points.
25. End of loop in non-sweep (\( \eta \)) direction.

Second ADI sweep, \( \eta \) direction

26. Set various sweep-dependent parameters, as follows:
   \[
   \begin{align*}
   \text{isweep} &= 2 \\
   \text{istep} &= \text{nlp} \\
   \text{del} &= \Delta \eta \\
   \text{npts} &= N_2 \text{ or } N_2 + 1 \\
   \text{nv} &= N_1 \text{ or } N_1 + 1
   \end{align*}
   \]
27. For axisymmetric flow, set \( J_1 = r/J \) at all grid points.
28. Set metrics in sweep ($\eta$) direction at all grid points as follows:

$$\text{metx}(i1,i2) = (\eta_{x})_{ij},$$
$$\text{mety}(i1,i2) = (\eta_{y})_{ij} \text{ or } (\eta_{z})_{ij},$$
$$\text{mett}(i1,i2) = (\eta_{t})_{ij}.$$

29. Begin loop in non-sweep ($\xi$) direction over interior points ($i = I1 = 2$ to $NPT1 - 1$).

30. For axisymmetric flow, set $\text{RAX}(i2) = r_{ij}$ along the current $\xi$-line at all $\eta$ grid points.

31. Call EQSTAT to get the derivatives of $p$ and $T$ with respect to $\rho$, $\rho u$, etc., along the current $\xi$-line at all $\eta$ grid points.

32. Call the COEF routines to compute the coefficients and source terms for the governing equations along the current $\xi$-line at all interior $\eta$ grid points.

33. End of loop in non-sweep ($\xi$) direction.

34. For axisymmetric flow, reset $\text{J1} = 1/J$ at all grid points.

35. For non-spatially periodic boundary conditions in the $\eta$ direction, begin loop in non-sweep ($\xi$) direction over interior points ($i = I1 = 2$ to $NPT1 - 1$).

36. Call EQSTAT to get the derivatives of $p$ and $T$ with respect to $\rho$, $\rho u$, etc., along the current $\xi$-line at all $\eta$ grid points.

37. Call BCGEN to compute the coefficients and source terms for the boundary condition equations at the end points ($j = I2 = 1$ and $N2$) of the current $\xi$-line.

38. If three-point boundary conditions were used at either boundary, call BCELM to eliminate the resulting off-diagonal coefficient submatrices.

39. End of loop in non-sweep ($\xi$) direction.

40. If implicit artificial viscosity is being used, call AVISCI to add the appropriate terms to the coefficient submatrices at all interior grid points.

41. If spatially periodic boundary conditions are being used in the $\eta$ direction, reset $\text{NPTS} = N2$.

42. Call ADI to solve the system of difference equations.

43. Begin loop in non-sweep ($\xi$) direction over interior points ($i = I1 = 2$ to $NPT1 - 1$).

44. Call UPDATE to compute the primitive flow variables $\rho^{n+1}$, $u^{n+1}$, etc., from the newly computed conservation variables in delta form, $\Delta \Omega$, along the current $\xi$-line at all $\eta$ grid points.

45. End of loop in non-sweep ($\xi$) direction.

**Finishing Steps**

46. If this is the first time step, reset the thin-layer flags back to their input value.

47. Call BVUP to update the $\xi$ boundary values, if necessary.

48. For all grid points, shift $\text{RHO}$ and $\text{RHOL}$ so that $\text{RHO} = \rho^{n+1}$ and $\text{RHOL} = \rho^n$. Similarly, shift the Fortran variables for $u$, $v$, $w$, and $E_T$. Finally, set $\text{TL} = T^n$. 

Proteus 2-D Programmer's Reference
Subroutine EXECT

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>TURBCH</td>
<td>BLK2</td>
<td>Manage solution of the $k$-$\varepsilon$ equations.</td>
</tr>
<tr>
<td></td>
<td>BLK2P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COEFS1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COEFS2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PERIOD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UPDTE</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

A

- CMUR
- CTHREE
- JI
- KBCPER
- NPT1, NPT2
- N1, N2
- N1P, N2P
- RHO
- S
- YPLUSD

Coefficient submatrix A.

Constant $C_\mu$, in formula for $C_\mu$.

Constant $C_3$ in formula for $C_\mu$.

Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.

Flags for spatially periodic boundary conditions in the $\xi$ and $\eta$ directions; 0 for non-periodic, 1 for periodic.

$N_1$ and $N_2$ for non-periodic boundary conditions, $N_1 + 1$ and $N_2 + 1$ for spatially periodic boundary conditions in $\xi$ and $\eta$.

Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.

Parameters specifying the dimension sizes in the $\xi$ and $\eta$ directions.

Static density $\rho$ at time level $n$.

Source term subvector $S$.

Nondimensional distance $y^+$ from the nearest solid wall.

**Output**

E, EL

ISWEEP

KE, KEL

MUT, MUTL

NPTS

NV

S

Turbulent dissipation rate $\varepsilon$ at time levels $n + 1$ and $n$.

Current ADI sweep number.

Turbulent kinetic energy $k$ at time levels $n + 1$ and $n$.

Turbulent viscosity $\mu_t$ at time levels $n + 1$ and $n$.

Number of grid points in the sweep direction, $N$.

Number of grid points in the "vectorized" direction, $N_v$.

Computed solution subvector $\Delta \hat{W}$.

**Description**

Subroutine EXECT manages the solution of the $k$-$\varepsilon$ equations. It is called by subroutine TURBCH, NTKE times per mean flow iteration. The steps involved in EXECT are described below.

**Preliminary Steps**

1. If spatially periodic boundary conditions are being used in either direction, call PERIOD to add the appropriate extra line(s) of data.
First ADI sweep, $\xi$ direction

2. Set various sweep-dependent parameters.
3. Call COEFS1 to compute the coefficients and source terms for the $k$-$\varepsilon$ equations.
4. Solve the system of difference equations by calling BLK2 for non-periodic boundary conditions, or BLK2P for periodic boundary conditions in the $\xi$ direction.

Second ADI sweep, $\eta$ direction

5. Set various sweep-dependent parameters.
6. Swap indices in the subvector S. The submatrix A is used as a temporary scratch array for this operation.
7. Call COEFS2 to compute the coefficients and source terms for the $k$-$\varepsilon$ equations.
8. Solve the system of difference equations by calling BLK2 for non-periodic boundary conditions, or BLK2P for periodic boundary conditions in the $\eta$ direction.

Finishing Steps

9. For all grid points, set KEL $= k^*$ and EL $= \varepsilon^*$.
10. Call UPDTKE to compute the primitive flow variables $k^{n+1}$ and $\varepsilon^{n+1}$ from $\Delta \hat{W}^n$, the newly computed conservation variables in delta form.
11. Compute the turbulent viscosity at each grid point, storing $\mu_t^{n+1}$ and $\mu_t^{n+1}$ in MUT and MUTL, respectively.
Subroutine FILTER (A,B,C,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLK3</td>
<td>BLKOUT</td>
<td>Rearrange rows of the boundary condition coefficient submatrices and</td>
</tr>
<tr>
<td>BLK4</td>
<td>ISAMAX</td>
<td>the source term subvector to eliminate any zeroes on the diagonal.</td>
</tr>
<tr>
<td>BLK5</td>
<td>ISRCHEQ</td>
<td></td>
</tr>
</tbody>
</table>

Input

- A, B, C
  - IDEBUG
  - IPR1TA, IPR2A
  - ISWEEP
  - IT
  - IV
  - NEQ
  - NOUT
  - NPRT1, NPRT2
  - NPTS
  - NVD, NPTSD
- S

Output

- A, B, C
- S

Description

Subroutine FILTER rearranges rows of the coefficient block submatrices and the source term subvector, at the two boundaries in the ADI sweep direction, in an attempt to eliminate any zero values on the diagonal of the submatrix B. These zero values may occur when mean flow boundary conditions are specified using the JBC and/or IBC input parameters, depending on the initial conditions and the order of the boundary conditions.

For instance, if the specified initial conditions are zero velocity and constant flow properties everywhere in the flow field, the perfect gas equation of state yields:

\[ E_T = \rho c_s T \]

\[ p = (\gamma - 1)E_T \]

\[ \frac{\partial p}{\partial \rho} = \frac{\partial p}{\partial \rho u} = \frac{\partial p}{\partial \rho v} = 0 \]

\[ \frac{\partial p}{\partial E_T} = \gamma - 1 \]
If, in addition, the boundary conditions at a given boundary are, in order, specified pressure $p = f$, no-slip $x$- and $y$-velocity $u = 0$ and $v = 0$, and specified temperature $T = f$, then the linearization of the boundary conditions leads to the following $B$ coefficient submatrix for that boundary:

\[
B = \begin{bmatrix}
0 & 0 & 0 & J(y - 1) \\
0 & J/\rho & 0 & 0 \\
0 & 0 & J/\rho & 0 \\
-JE_T/c_\rho^2 & 0 & 0 & J/c_\rho \\
\end{bmatrix}
\]

The zero on the diagonal will lead to a divide-by-zero error in subroutine BLK4, even though this is not a singular matrix.

Subroutine FILTER tries to fix this problem. In this example, it finds a zero at element $B_{11}$, searches column 1 for the largest element in absolute value (in this case $-JE_T/c_\rho^2$), and adds that row to the row with the zero on the diagonal. Of course, the corresponding rows of $A$, $C$, and $S$ must also be added together. The new $B$ submatrix would be:

\[
B = \begin{bmatrix}
-JE_T/c_\rho^2 & 0 & 0 & J(y - 1) + J/c_\rho \\
0 & J/\rho & 0 & 0 \\
0 & 0 & J/\rho & 0 \\
-JE_T/c_\rho^2 & 0 & 0 & J/c_\rho \\
\end{bmatrix}
\]

**Remarks**

1. If a column with a zero on the diagonal has no other elements greater than $10^{-10}$, then it is assumed that the matrix $B$ is singular, which means the specified boundary conditions are not independent of one another. An error message is printed and the calculation is stopped.

2. It's probably sufficient to only call this subroutine for the first time step. After the first step, the chances of $u$ and $v$ both being exactly zero at the same interior grid point are slim. Nevertheless, in the current version of Proteus, FILTER is called at every time step.

3. The Cray search routine ISAMAX is used in finding the largest element in any column corresponding to a zero on the matrix diagonal. The Cray search routine ISRCHEQ is used in determining the grid locations for debug printout.

4. This subroutine generates the output for the IDEBUG(4) option.
Subroutine FTEMP

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITC</td>
<td></td>
<td>Compute auxiliary variables that are functions of temperature.</td>
</tr>
<tr>
<td>MAIN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- CCP1, CCP2, CCP3, CCP4: Constants in formula for specific heat.
- CK1, CK2: Constants in formula for laminar thermal conductivity coefficient.
- CMU1, CMU2: Constants in formula for laminar viscosity coefficient.
- *GAMR*: Reference ratio of specific heats, \( \gamma \).
- IGAM: Flag for constant or variable \( c_p \), \( c_v \), and \( \gamma \); 0 if they are to be computed as functions of temperature, 1 if they are to be treated as constant.
- *ILAMV*: Flag for computation of laminar viscosity and thermal conductivity.
- *NOUT*: Unit number for standard output.
- *N1, N2*: Number of grid points \( N_1 \) and \( N_2 \) in the \( \xi \) and \( \eta \) directions.
- RGAS: Gas constant \( R \).
- T: Static temperature \( T \).
- *TR, UR, MUR, KTR*: Reference temperature \( T_r \), velocity \( u_\tau \), viscosity \( \mu_\tau \), and thermal conductivity \( k_\tau \).

**Output**

- CP, CV: Specific heats \( c_p \) and \( c_v \).
- MU, LA, KT: Laminar coefficient of viscosity \( \mu_\tau \), laminar second coefficient of viscosity \( \lambda_\tau \), and laminar coefficient of thermal conductivity \( k_\tau \).

**Description**

Subroutine FTEMP computes the auxiliary variables \( \mu_\tau \), \( \lambda_\tau \), \( k_\tau \), \( c_p \), and \( c_v \). For the laminar viscosities \( \mu_\tau \) and \( \lambda_\tau \), and the laminar thermal conductivity \( k_\tau \), there are two options currently available.

If the input parameter ILAMV = 0 (the default), FTEMP sets the nondimensional values as:

\[
\begin{align*}
\mu_i &= 1 \\
\lambda_i &= -2/3 \\
k_i &= 1
\end{align*}
\]

Thus, with this option, the laminar viscosity and thermal conductivity are held constant at their reference values. These reference values may be specified by the user, or computed from the reference temperature. The laminar second coefficient of viscosity \( \lambda_i \) is set equal to \(-2\mu_i/3\).

If ILAMV = 1, \( \mu_i \) and \( k_i \) are computed as functions of temperature using Sutherland's formula (White, 1974). The formula for the laminar viscosity coefficient \( \mu_i \) is
\[ \mu_t = \frac{\mu}{\mu_r} = \frac{\mu'_r}{\mu_r} \frac{T_r + C\mu_2}{T} \left( \frac{T}{T_r} \right)^{3/2} \]

where the overbar indicates a dimensional value, and \( \mu'_r \) is the laminar viscosity coefficient at \( T' = T_r \), given by

\[ \mu'_r = C\mu_1 \frac{T_r^{3/2}}{T + C\mu_2} \]

Depending on the namelist input values of MUR and RER, \( \mu'_r \) may or may not be equal to \( \mu_r \). These formulas are valid for air for temperatures from 300 to 3420 °R (167 to 1900 K). The value of the constants \( C\mu_1 \) and \( C\mu_2 \) depend on whether reference values are being specified by the user in English units (\( \text{IUNITS} = 0 \)) or SI units (\( \text{IUNITS} = 1 \)). The values are presented in Table 4-1. The laminar second coefficient of viscosity \( \lambda_l \) is set equal to \(-2\mu'_r/3\). The formula for the laminar thermal conductivity coefficient \( k_l \) is

\[ k_l = \frac{k'_l}{k_r} = \frac{k'_r}{k_r} \frac{T_r + Ck_2}{T} \left( \frac{T}{T_r} \right)^{3/2} \]

where the overbar indicates a dimensional value, and \( k'_r \) is the laminar thermal conductivity coefficient at \( T' = T_r \), given by

\[ k'_r = Ck_1 \frac{T_r^{3/2}}{T + Ck_2} \]

Depending on the namelist input values of KTR and PRLR, \( k'_r \) may or may not be equal to \( k_r \). These formulas are valid for air for temperatures from 300 to 1800 °R (167 to 1000 K). The value of the constants \( Ck_1 \) and \( Ck_2 \) depend on whether reference values are being specified by the user in English units (\( \text{IUNITS} = 0 \)) or SI units (\( \text{IUNITS} = 1 \)). The values are presented in Table 4-1.

There are also two options available for the specific heat coefficients \( c_p \) and \( c_v \). If the flag \( \text{IGAM} = 1 \), a value of the specific heat ratio \( \gamma \) has been specified by the user. In this case \( c_p \) and \( c_v \) are treated as constants, and computed from

\[ c_v = \frac{R}{\gamma - 1} \]
\[ c_p = c_v + R \]

If \( \text{IGAM} = 0 \), the user did not specify a value of \( \gamma \). In this case, the specific heat coefficient \( c_p \) is computed as a function of temperature from the empirical formula of Hesse and Mumford (1964), and \( c_v \) is computed from that value assuming a thermally perfect gas. The ratio \( \gamma = c_p/c_v \) will then vary with temperature. The equations for \( c_p \) and \( c_v \) are:

\[ c_p = \bar{c}_p \frac{T_r}{u'_{\infty}} = \frac{T_r}{u'_{\infty}} \left( C_{c_p1} - C_{c_p2} T^{-1/2} - C_{c_p3} T - C_{c_p4} \bar{T}' \right) \]
\[ c_v = c_p - R \]

This formula is valid for air for temperatures from 540 to 9000 °R (300 to 5000 K). The values of the constants \( C_{c_p1} \) through \( C_{c_p4} \) are presented in Table 4-1.
TABLE 4-1. EMPIRICAL CONSTANTS FOR $\mu$, $k$, AND $c_p$

<table>
<thead>
<tr>
<th>CONSTANT</th>
<th>ENGLISH UNITS</th>
<th>SI UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{o1}$</td>
<td>$7.3035 \times 10^{-7}$</td>
<td>$1.4582 \times 10^{-6}$</td>
</tr>
<tr>
<td>$C_{o2}$</td>
<td>198.6</td>
<td>110.3</td>
</tr>
<tr>
<td>$C_{o3}$</td>
<td>$7.4907 \times 10^{-3}$</td>
<td>$1.8641 \times 10^{-3}$</td>
</tr>
<tr>
<td>$C_{o4}$</td>
<td>350.0</td>
<td>194.4</td>
</tr>
<tr>
<td>$C_{mp1}$</td>
<td>$8.53 \times 10^3$</td>
<td>$1.4264 \times 10^3$</td>
</tr>
<tr>
<td>$C_{mp2}$</td>
<td>$3.12 \times 10^4$</td>
<td>$3.8888 \times 10^5$</td>
</tr>
<tr>
<td>$C_{mp3}$</td>
<td>$2.065 \times 10^6$</td>
<td>$1.9184 \times 10^5$</td>
</tr>
<tr>
<td>$C_{mp4}$</td>
<td>$7.83 \times 10^8$</td>
<td>$4.0413 \times 10^7$</td>
</tr>
</tbody>
</table>

Remarks

1. The formulas used with the ILAMV = 1 option are for air. For other fluids, different formulas should be used to compute $\mu$, $\lambda$, and $k$. These could easily be programmed as additional ILAMV options. Or, if the flow being computed is such that $\mu$ and $k$ may be considered constant, simply set ILAMV = 0 and read in the appropriate values for $\mu$ and $k$. Note, however, that the ILAMV = 0 option still sets $\lambda = -2\mu/3$.

2. The formula used to compute $c_p$ when a value of $\gamma$ is not specified by the user, is for air. For other gases, a different formula should be programmed. Or, if $c_p$ and $c_v$ may be considered constant, a value of $\gamma$ should be read in.

3. An error message is generated and execution is stopped if an illegal value is specified for ILAMV.
Subroutine GEOM

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>METS</td>
<td>Manage computation of grid and metric parameters.</td>
</tr>
<tr>
<td></td>
<td>PAK</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- IPACK  
  Flags for grid packing option.
- NGEOM  
  Flag for type of computational coordinates.
- NGRID  
  Unit number for input mesh file.
- NOUT   
  Unit number for standard output.
- N1, N2  
  Number of grid points \( N_i \) and \( N_j \), in the \( \xi \) and \( \eta \) directions.
- N1P, N2P  
  Parameters specifying the dimension sizes in the \( \xi \) and \( \eta \) directions.
- RMIN, RMAX  
  Minimum and maximum \( r' \)-coordinates for polar grid.
- THMIN, THMAX  
  Minimum and maximum \( \theta' \)-coordinates for polar grid.
- XMN, XMAX  
  Minimum and maximum \( x \)-coordinates for Cartesian grid.
- YMIN, YMAX  
  Minimum and maximum \( y \)-coordinates for Cartesian grid.

**Output**

- DXI, DETA  
  Computational grid spacing \( \Delta_{\xi} \) and \( \Delta_{\eta} \).
- X, Y  
  Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \).

**Description**

Subroutine GEOM manages the computation of the grid and metric parameters. There are currently three coordinate system options built into Proteus, as follows:

<table>
<thead>
<tr>
<th>NGEOM</th>
<th>Computational Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cartesian ((x,y))</td>
</tr>
<tr>
<td>2</td>
<td>Polar ((r',\theta'))</td>
</tr>
<tr>
<td>10</td>
<td>Read from separate file.</td>
</tr>
</tbody>
</table>

Subroutine GEOM first computes the grid spacing in computational space in the \( \xi \) and \( \eta \) directions as \( \Delta_{\xi} = 1/(N_1 - 1) \) and \( \Delta_{\eta} = 1/(N_2 - 1) \). Note that grid points in computational space are always evenly distributed along the \((\xi,\eta)\) coordinate lines.

**Cartesian \((x,y)\) Coordinates \((NGEOM = 1)\)**

For the Cartesian coordinate option, an evenly spaced set of physical Cartesian \((x,y)\) coordinates are related to the computational \((\xi,\eta)\) coordinates by

\[
x = x_{\text{min}} + (x_{\text{max}} - x_{\text{min}})\frac{\xi}{\Delta_{\xi}}
\]

\[
y = y_{\text{min}} + (y_{\text{max}} - y_{\text{min}})\frac{\eta}{\Delta_{\eta}}
\]

These equations also apply to axisymmetric flow, with \( y \) representing the radius \( r \). If grid packing is used, subroutine PAK is called to redistribute these points according to the packing parameters specified by the
user, and to interpolate to get the new physical Cartesian (x-y) coordinates in the computational mesh. Subroutine METS is then called to numerically compute the grid transformation metrics and Jacobian.

**Polar (r'-θ') Coordinates (NGEOM = 2)**

For the polar coordinate option, an evenly spaced set of physical polar (r'-θ') coordinates are related to the computational (ξ-η) coordinates by

\[
\theta' = \theta'_{\text{min}} + (\theta'_{\text{max}} - \theta'_{\text{min}}) \xi \\
r' = r'_{\text{min}} + (r'_{\text{max}} - r'_{\text{min}}) \eta
\]

The Cartesian (x-y) coordinates are simply given by

\[
x = r' \cos \theta' \\
y = r' \sin \theta'
\]

The above equations also could be used in axisymmetric flow, with \( y \) representing the radius \( r \). As in the NGEOM = 1 option, if grid packing is used, subroutine PAK is called to redistribute these points according to the packing parameters specified by the user, and to interpolate to get the new physical Cartesian (x-y) coordinates in the computational mesh. Subroutine METS is then called to numerically compute the grid transformation metrics and Jacobian.

**Coordinates Read From Separate File (NGEOM = 10)**

The third option for specifying the computational coordinate system is to read it from a separate file, as described in Section 3.2 of Volume 2. The computational (ξ-η) coordinate system is determined by a set of \( N_{G1} \times N_{G2} \) points whose physical Cartesian (x-y) coordinates are specified. Here \( N_{G1} \) and \( N_{G2} \) are the number of points in the \( \xi \) and \( \eta \) directions used to specify the computational coordinate system. Note that they do not have to be equal to \( N_1 \) and \( N_2 \), the number of points in the computational mesh used for the finite-difference method. Note also that the points do not have to be equally distributed in physical space along the \( \xi \) and \( \eta \) coordinate lines.

If grid packing is being used, subroutine PAK is called to distribute \( N_1 \times N_2 \) computational mesh points in physical space according to the packing parameters SQ specified by the user, and to interpolate among the \( N_{G1} \times N_{G2} \) points in the input computational coordinate system to get the new physical Cartesian coordinates of the points in the computational mesh.

If grid packing is not being used, but \( N_{G1} \) and \( N_{G2} \) are not equal to \( N_1 \) and \( N_2 \) respectively, then subroutine PAK is still called. In this case, however, PAK distributes the \( N_1 \times N_2 \) computational mesh points evenly in physical space and then interpolates among the \( N_{G1} \times N_{G2} \) points in the input computational coordinate system to get the new physical Cartesian coordinates of the points in the computational mesh.

In either case, subroutine METS is then called to numerically compute the grid transformation metrics and Jacobian.

**Remarks**

1. There may be some confusion between the axisymmetric flow option and the polar coordinate system option, or between the axisymmetric radius \( r \) and the polar coordinate \( r' \). They are not the same thing. The governing flow equations were originally developed by writing them in Cartesian (x-y) coordinates, then transforming them into generalized (ξ-η) coordinates. Therefore, any computational coordinate system that is used, including the polar coordinate system, must be related to the original Cartesian system through the transformation metrics and Jacobian. The parameters \( r' \) and \( \theta' \) are used only to initially define the coordinates in the NGEOM = 2 option. Now, if the (x-y) coordinates, no matter how they are obtained, are rotated about the Cartesian x axis, the result is a cylindrical coordinate co-

---

26 The distinction between the computational coordinate system and the computational mesh is described in Section 2.2 of Volume 2.
ordinate system with $y$ representing the radius $r$. Thus, the axisymmetric flow option can be used with any of the coordinate system options. The polar coordinate option would be useful, for instance, for flow over a sphere.

2. An error message is generated and execution is stopped if an illegal coordinate system option is specified.

3. With the NGEOM = 10 option, an error message is generated and execution is stopped if $N_{o1}$ and or $N_{o2}$ are greater than the dimensioning parameters N1P and or N2P.
Subroutine INIT

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITC</td>
<td></td>
<td>Get user-defined initial flow field.</td>
</tr>
</tbody>
</table>

**Input**

- ICVARS
  - Flag specifying which variables are being supplied as initial conditions by subroutine INIT.
- NIN
  - Unit number for namelist input.
- NOUT
  - Unit number for standard output.
- N1, N2
  - Number of grid points N1 and N2, in the \( \xi \) and \( \eta \) directions.

**Output**

\( \mathbf{P}, \mathbf{T}, \mathbf{U}, \mathbf{V}, \mathbf{W} \)

- Initial flow field values of static pressure \( p \), static temperature \( T \), and velocities \( u, v, \) and \( w \).

**Description**

Subroutine INIT supplies the user-defined initial flow field. In general, this subroutine will be tailored to the problem being solved, and supplied by the user. Details on the variables to be supplied by INIT are presented in Section 5.1 of Volume 2.

A default version of INIT is supplied with Proteus that specifies uniform flow with constant properties everywhere in the flow field. The above list of input and output Fortran variables are for the default version of INIT. The default version assumes ICVARS = 2 (the default value), and reads values of \( p_0, u_0, v_0, w_0, \) and \( T_0 \) from namelist IC. The defaults for these parameters are 1.0, 0.0, 0.0, 0.0, and 1.0, respectively, resulting in an initial flow field with \( \bar{p} = p, u = v = w = 0 \), and \( \bar{T} = T_0 \).

**Remarks**

1. If a value for ICVARS other than 2 is set in the input, a warning message is generated and ICVARS is reset to 2.
2. Subroutine INIT is a convenient place to specify point-by-point boundary condition types and values. It’s often easier to do this using Fortran coding rather than entering each value into the namelist input file.
### Subroutine INITC

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>EQSTAT</td>
<td>Set up consistent initial conditions based on data from INIT.</td>
</tr>
<tr>
<td></td>
<td>FTEMP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KEINIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TURBBL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YPLUSN</td>
<td></td>
</tr>
</tbody>
</table>

### Input

- **CMUR**  
  Constant $C_m$, in formula for $C_a$.
- **CTHREE**  
  Constant $C_3$ in formula for $C_u$.
- **GAMR**  
  Reference ratio of specific heats, $\gamma_r$.
- **GC**  
  Proportionality factor $g$, in Newton’s second law.
- **HSTAG**  
  Stagnation enthalpy $h_r$ used with constant stagnation enthalpy option.
- **ICVARS**  
  Flag specifying which variables are being supplied as initial conditions by subroutine INIT.
- **IHSTAG**  
  Flag for constant stagnation enthalpy option.
- **IREST**  
  Flag for reading restart file.
- **ITBEG**  
  The time level $n$ at the beginning of a run.
- **ITURB**  
  Flag for turbulent flow option.
- **KBC1, KBC2**  
  Boundary types for the $\xi$ and $\eta$ directions.
- **LWSET**  
  Flags specifying how wall locations are to be determined for the turbulence model; 0 if wall locations are to be found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters.
- **N1, N2**  
  Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.
- **PR**  
  Reference pressure $p_r$.
- **RGAS**  
  Gas constant $R$.
- **RHOR, UR**  
  Reference density $\rho$, and velocity $u_r$.

**INITIAL FLOW FIELD**  
From the user-supplied or default version of subroutine INIT. The combination of variables supplied by INIT is specified by ICVARS. See Section 5.0 of Volume 2 for details.

### Output

- **LWALL1, LWALL2**  
  Flags specifying wall locations for $\xi$ and $\eta$ boundaries, if not set in input.
- **MUT, MUTL**  
  Turbulent viscosity $\mu_t$ at time levels $n$ and $n - 1$.
- **RHO, U, V, W, ET**  
  Initial flow field values of static density $\rho$, velocities $u$, $v$, and $w$, and total energy $E_T$ at time level $n$.  

---

Proteus 2-D Programmer’s Reference 4.0 Proteus Subprograms: INITC 199
RHOL, UL, VL, WL, ETL  
Initial flow field values of static density $\rho$, velocities $u$, $v$, and $w$, and total energy $E_T$ at time level $n-1$.  

TL  
Static temperature $T$ at time level $n-1$.  

**Description**  
Subroutine INITC sets up consistent initial flow field conditions based on the data supplied by subroutine INIT. For restart cases, subroutine REST is called to read the computational mesh and the initial flow field. Otherwise, the data supplied by INIT are used to obtain the density $\rho$, the velocities $u$, $v$, and $w$, and the temperature $T$. It then calls FTEMP to compute the laminar viscosity coefficients $\mu_1$ and $\lambda_1$, the laminar thermal conductivity coefficient $k_l$, and the specific heat coefficients $c_p$ and $c_v$. EQSTAT is called next to compute the pressure $p$ and to recompute the temperature $T$. For turbulent flow, the appropriate subroutines are called to compute the effective viscosity and thermal conductivity coefficients using the turbulence model specified by the user. And finally, for non-restart cases, the values of the dependent variables at time level $n-1$ are set equal to the values at level $1$.  

The flag ICVARS is used to specify which combination of variables are being supplied by INIT. The calculation of $\rho$, $u$, $v$, $w$, and $T$ is described below for the different values of ICVARS. In all of the equations below, the specific heats are defined by  

$$c_v = \frac{R}{\gamma - 1}$$  

$$c_p = R + c_v$$  

where $\gamma$ is either specified by the user or computed from the reference temperature $T_r$.  

**ICVARS = 1**  

With this option, the density $\rho$, the momentum components $\rho u$, $\rho v$, and $\rho w$, and if IHSTAG = 0 the total energy $E_T$, are supplied by INIT. Thus, the velocity components are simply  

$$u = \frac{\rho u}{\rho}$$  

$$v = \frac{\rho v}{\rho}$$  

$$w = \frac{\rho w}{\rho}$$  

If the energy equation is being solved (IHSTAG = 0), the temperature is computed from  

$$T = \frac{1}{c_v} \left[ \frac{E_T}{\rho} - \frac{1}{2} (u^2 + v^2 + w^2) \right]$$  

If the energy equation is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from  

$$T = \frac{1}{c_p} \left[ h_T - \frac{1}{2} (u^2 + v^2 + w^2) \right]$$  

27 The calculation of $T$ at this point may be approximate. See Remark 1.  

28 See Remark 1.
ICVARS = 2

With this option, the pressure \( p \) and the velocities \( u, v, \) and \( w \) are supplied by \textsc{INIT}. If the energy equation is being solved (IHSTAG = 0), the temperature \( T \) is also supplied by \textsc{INIT}. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

\[
T = \frac{1}{c_p} \left[ h_T - \frac{1}{2} (u^2 + v^2 + w^2) \right]
\]

The density is then given by

\[
\rho = \frac{p}{RT}
\]

and the total energy is

\[
E_T = \rho \left[ c_v T + \frac{1}{2} (u^2 + v^2 + w^2) \right]
\]

ICVARS = 3

With this option, the density \( \rho \) and the velocities \( u, v, \) and \( w \) are supplied by \textsc{INIT}. If the energy equation is being solved (IHSTAG = 0), the temperature \( T \) is also supplied by \textsc{INIT}. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

\[
T = \frac{1}{c_p} \left[ h_T - \frac{1}{2} (u^2 + v^2 + w^2) \right]
\]

The total energy is then

\[
E_T = \rho \left[ c_v T + \frac{1}{2} (u^2 + v^2 + w^2) \right]
\]

ICVARS = 4

With this option, the pressure \( p \) and the velocities \( u, v, \) and \( w \) are supplied by \textsc{INIT}. If the energy equation is being solved (IHSTAG = 0), the density \( \rho \) is also supplied by \textsc{INIT}. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), this option is the same as the ICVARS = 2 option. If the energy equation is being solved, then, the temperature is

\[
T = \frac{p}{\rho R}
\]

The total energy is then

\[
E_T = \rho \left[ c_v T + \frac{1}{2} (u^2 + v^2 + w^2) \right]
\]

ICVARS = 5

With this option, the static pressure coefficient \( c_p \) and the velocities \( u, v, \) and \( w \) are supplied by \textsc{INIT}. If the energy equation is being solved (IHSTAG = 0), the temperature \( T \) is also supplied by \textsc{INIT}. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

\[
T = \frac{1}{c_p} \left[ h_T - \frac{1}{2} (u^2 + v^2 + w^2) \right]
\]

The pressure coefficient is defined by
The nondimensionalized pressure \( p = \frac{\bar{p} - p_i}{g_c/\rho u^2} \) is thus

\[
p = \frac{c_p}{2} + \frac{p_i g_c}{\rho u^2}
\]

or, since \( p_i = \rho_i R T_i / g_c \) and the nondimensionalized gas constant \( R = \bar{R} T_i / u^2 \),

\[
p = \frac{c_p}{2} + R
\]

The density is then

\[
\rho = \frac{p}{RT}
\]

and the total energy is

\[
E_T = \rho \left[ c_p T + \frac{1}{2} (u^2 + v^2 + w^2) \right]
\]

**ICVARS = 6**

With this option, the pressure \( p \), Mach number \( M \), and flow angles \( \alpha \), and \( \omega \) are supplied by INIT. If the energy equation is being solved (IHSTAG = 0), the temperature \( T \) is also supplied by INIT. If it is being eliminated by assuming constant stagnation enthalpy (IHSTAG = 1), the temperature is computed from

\[
T = T_T \left( 1 + \frac{\gamma_r - 1}{2} M^2 \right)^{-1}
\]

where \( T_T = h_T / c_p \). The density is

\[
\rho = \frac{p}{RT}
\]

The flow angles are defined by \( \alpha_v = \tan^{-1}(v/u) \) and \( \alpha_w = \tan^{-1}(w/u) \). The Mach number is defined by

\[
M = \left( \frac{u^2 + v^2 + w^2}{\gamma_r RT} \right)^{1/2}
\]

Solving for \( u \),

\[
u = M \left( \frac{\gamma_r RT}{1 + (v/u)^2 + (w/u)^2} \right)^{1/2}
\]

where \( (v/u)^2 = \tan^2 \alpha_v \) and \( (w/u)^2 = \tan^2 \alpha_w \). The remaining velocities are simply

\[
v = u \tan \alpha_v
\]

\[
w = u \tan \alpha_w
\]

The total energy is
\[ E_T = \rho \left[ c_v T + \frac{1}{2} (u^2 + v^2 + w^2) \right] \]

**Remarks**

1. If \( T \) is not supplied by INIT, it must be computed from the equation of state. The equation of state contains a specific heat coefficient (either \( c_p \) or \( c_v \), depending on whether the stagnation enthalpy is assumed constant or not.) The first time \( T \) is computed in INITC, a constant value of specific heat is used, consistent with the reference temperature \( T_r \). If the user specified constant specific heat (i.e., a value for \( y \), was read in), this is not a problem. However, if the temperature-dependent specific heat option is being used (i.e., a value for \( y \) was not read in), the equation of state and the empirical equation for specific heat are coupled. For this reason \( T \) is recomputed in EQSTAT after the specific heats are computed in FTEMP. Ideally, this coupling would be handled by iteration between FTEMP and EQSTAT. This is not currently done in *Proteus*, however.

2. For options in which the pressure \( p \) is specified (ICVARS = 2, 4, and 6), the value supplied by INIT is redefined as follows:

\[
p = \frac{p_r \gamma_c}{\rho \mu^2}
\]

This is necessary because input and output values of \( p \) are nondimensionalized by the reference pressure \( p_r = \rho_r R T_r \), while internal to the code itself \( p \) is nondimensionalized by the normalizing pressure \( p_n = \rho_n \mu^2 \). See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.

3. With the ICVARS = 6 option, the initial velocity \( u \) will be limited to non-negative values.

4. If non-positive pressures or temperatures were computed in EQSTAT, the Fortran variable INEG will be positive. An error message will be printed, including a table showing the location of the non-positive values. The calculation will stop in INITC.

5. An error message is generated and execution is stopped if an illegal value is specified for ICVARS.

6. An error message is generated and execution is stopped if the value of ITURB does not correspond to an existing turbulence model.
Subroutine INPUT

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>ISAMAX</td>
<td>Read and print input, perform various initializations.</td>
</tr>
</tbody>
</table>

### Input

- **NIN**: Unit number for namelist input.
- **NTP**: Dimensioning parameter specifying the maximum number of entries in the table of time-dependent boundary condition values.
- **NTSEQP**: Dimensioning parameter specifying the maximum number of time step sequences for the time step sequencing option.
- **N1P, N2P**: Parameters specifying the dimension sizes in the $\xi$ and $\eta$ directions.

### Output

- **CKMIN**: Constant $(C_{k\text{ee}})_{\text{ns}}$ in the Klebanoff intermittency factor.
- **GAMR**: Reference ratio of specific heats, $\gamma_r$.
- **HSTAG, HSTAGR**: Dimensionless and dimensional stagnation enthalpy $h_r$ for the constant stagnation enthalpy option.
- **IGAM**: Flag for constant or variable $c_r$, $c_v$, and $\gamma_r$; 0 if they are to be computed as functions of temperature, 1 if they are to be treated as constant.
- **IPRT1A, IPRT2A**: Indices for printout in the $\xi$ and $\eta$ directions.
- **ITDBC**: Flag for time-dependent boundary conditions; 0 if all boundary conditions are steady, 1 if any general unsteady boundary conditions are used, 2 if only steady and time-periodic boundary conditions are used.
- **LWALL1, LWALL2**: Flags specifying wall locations for $\xi$ and $\eta$ boundaries.
- **LWSET**: Flags specifying how wall locations are to be determined for the turbulence model; 0 if wall locations are to be found automatically by searching for boundary points where the velocity is zero, 1 if input using the LWALL parameters, 2 if input using the IWALL parameters.
- **MACHR**: Reference Mach number $M_r$.
- **MUR, KTR**: Reference viscosity coefficient $\mu_r$ and thermal conductivity coefficient $k_r$.
- **NEQ**: Number of coupled equations being solved, $N_e$.
- **NPRT1, NPRT2**: Total number of indices for printout in the $\xi$ and $\eta$ directions.
- **NRW, NET**: Array indices associated with the dependent variables $\rho w$ and $E_r$.
- **NZM, NEN**: Array indices associated with the swirl momentum and energy equations.
- **PR**: Reference pressure $p_r$.
- **PRLR**: Reference laminar Prandtl number $Pr_l$.
- **RER, PRR**: Reference Reynolds number $Re_r$ and Prandtl number $Pr_r$.
Subroutine INPUT performs various input and initialization functions. It first reads the title and namelist input from the standard input file. Namelist RSTRT is read first, followed by namelist IO. If IUNITS = 1, indicating reference conditions will be specified in SI units, various default values and constants are redefined to be consistent with SI units. The remaining namelists are then read.

Next, the flags controlling the time step cycling and the convergence testing method are redefined, if necessary, to be consistent with each other. The number of equations being solved, and the array indices corresponding to the energy and swirl momentum equations, are then determined based on the values of IHSTAG and ISWRL. A flag is set if time-dependent boundary conditions are being used. If the thin-layer option is being used, the flags ITXI and ITETA used in the Baldwin-Lomax turbulence model are automatically set equal to values consistent with the thin-layer approximation. The LWSET flags, which specify how wall locations are to be determined for the turbulence model, are defined based on the default and input values of the LWALL and IWALL parameters. If the user did not specify a value for \( (C_{kh})_{min} \), it is set to the default value, which depends on the turbulence model being used.

Next, if frequency of printout in the \( \xi \) and \( \eta \) directions is being set by the input arrays IPRT1 and IPRT2, the corresponding grid indices are stored in arrays IPRT1A and IPRT2A. The total number of printout locations in each direction is also determined.

A header is then written to the standard output file, followed by the input namelists. Note that, for variables not specified by the user in the input namelists, the values in this printout will be the default values.

Various checks are made for inconsistent or invalid input, and appropriate error or warning messages are written. These are described in Section 7.0 of Volume 2.

Next, any reference or normalizing conditions not already defined are calculated. The reference and normalizing conditions are then written to the standard output file, with the appropriate units. See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.

Remarks

1. The Cray search routine ISAMAX is used in the input consistency check to determine whether any implicit artificial viscosity coefficients are non-zero.
Function ISAMAX (N,V,INC)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOUT1</td>
<td></td>
<td>Find the first index corresponding to the largest absolute value of the elements of a Fortran vector.</td>
</tr>
<tr>
<td>BLOUT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMSTP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- **N**: Number of elements to process in the vector (i.e., \( N = \) vector length if \( INC = 1 \), \( N = \) (vector length)/2 if \( INC = 2 \), etc.).
- **V**: Vector to be searched.
- **INC**: Skip distance between elements of V. For contiguous elements, \( INC = 1 \).

**Output**

- **ISAMAX**: First index corresponding to the largest absolute value of the elements of V that were searched.

**Description**

Function ISAMAX finds the first index corresponding to the largest absolute value of the elements of a vector. For a one-dimensional vector, the use of ISAMAX is straightforward. For example,

\[
imax = \text{isamax}(np, v, 1)
\]

sets \( IMAX \) equal to the index \( I \) corresponding to the maximum value of \( V(I) \) for \( I = 1 \) to NP.

A starting location can be specified, as in

\[
imax = 4 + \text{isamax}(np-4, v(5), 1)
\]

which sets \( IMAX \) equal to the index \( I \) corresponding to the maximum value of \( V(I) \) for \( I = 5 \) to NP.

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if \( A \) is an array dimensioned \( \text{NDIM1} \) by \( \text{NDIM2} \), then

\[
imax = \text{isamax}(\text{NDIM1}\times\text{NDIM2}, a, 1)
\]

sets \( IMAX \) equal to the one-dimensional index corresponding to the maximum value of \( A(I,J) \) for all \( I \) and \( J \). The maximum value of \( A \) can then be referenced as \( A(IMAX,1) \).

One dimension at a time can also be searched. For example,

\[
imax = \text{isamax}(\text{NDIM1}, a(1,5), 1)
\]

sets \( IMAX \) equal to the index \( I \) corresponding to the maximum value of \( A(I,5) \) for \( I \) varying from 1 to \( \text{NDIM1} \). Similarly, by specifying a skip increment,
\( jmax = \text{isamax}(\text{ndim2}, a(5,j), \text{ndim1}) \)

sets JMAX equal to the index J corresponding to the maximum value of \( A(5,J) \) for J varying from 1 to NDIM2.

**Remarks**

1. ISAMAX is a Cray search routine (Cray Research, Inc., 1989b).
Function ISAMIN (N,V,INC)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOUT1</td>
<td></td>
<td>Find the first index corresponding to the smallest absolute value of the elements of a Fortran vector.</td>
</tr>
<tr>
<td>BLOUT2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- **N**: Number of elements to process in the vector (i.e., \( N = \text{vector length if INC} = 1, \ N = (\text{vector length})/2 \text{ if INC} = 2, \) etc.).
- **V**: Vector to be searched.
- **INC**: Skip distance between elements of V. For contiguous elements, INC = 1.

**Output**

- **ISAMIN**: First index corresponding to the smallest absolute value of the elements of V that were searched.

**Description**

Function ISAMIN finds the first index corresponding to the smallest absolute value of the elements of a vector. It is used in exactly the same way as ISAMAX.

**Remarks**

1. ISAMIN is a Cray search routine (Cray Research, Inc., 1989b).
Function ISRCHEQ (N,V,INC,VALUE)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCGEN</td>
<td></td>
<td>Find the first index in a vector whose element is equal to a specified value.</td>
</tr>
<tr>
<td>BLIN1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLIN2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOUT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOUT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILTER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input

- **N**
  - Number of elements to process in the vector (i.e., N = vector length if INC = 1, N = (vector length)/2 if INC = 2, etc.).
- **V**
  - Vector to be searched.
- **INC**
  - Skip distance between elements of V. For contiguous elements, INC = 1.
- **VALUE**
  - Value to be searched for in the vector V.

Output

- **ISRCHEQ**
  - First index, of the elements of V that were searched, whose element is equal to the value V. If the value V is not found, the returned value will be N + 1.

Description

Function ISRCHEQ finds the first index in a vector whose element is equal to a specified value. For a one-dimensional vector, the use of ISRCHEQ is straightforward. For example,

\[
\text{ival} = \text{isrcheq}(\text{np},v,1,\text{val})
\]

searches V(I), for I = 1 to NP, for the value VAL, and sets IVAL equal to the first index I for which V(I) = VAL. If the value VAL is not found, IVAL will be equal to NP + 1.

A starting location can be specified, as in

\[
\text{ival} = 4 + \text{isrcheq}(\text{np}-4,v(5),1,\text{val})
\]

which searches V(I), for I = 5 to NP, for the value VAL.

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if A is an array dimensioned NDIM1 by NDIM2, then

\[
\text{ival} = \text{isrcheq}(\text{ndim1}*\text{ndim2},a,1,\text{val})
\]

searches A(I,J), for all I and J, for the value VAL, and sets IVAL equal to the corresponding one-dimensional index. The desired indices in A can then be recovered from

\[
i = \text{mod(ival-1,ndim1)} + 1
\]

\[
j = (\text{ival-1})/\text{ndim1} + 1
\]
One dimension at a time can also be searched. For example,

$$i_{val} = \text{isrcheq}(\text{ndim1}, a(1,5), 1, \text{val})$$

searches $A(I,5)$, for $I = 1$ to $\text{NDIM1}$, for the value $\text{VAL}$. Similarly, by specifying a skip increment,

$$j_{val} = \text{isrcheq}(\text{ndim2}, a(5,j), \text{ndim1}, \text{val})$$

searches $A(5,J)$, for $J = 1$ to $\text{NDIM2}$, for the value $\text{VAL}$.

**Remarks**

1. ISRCEQ is a Cray search routine (Cray Research, Inc., 1989b).
Subroutine KEINIT

Called by | Calls | Purpose
----------|-------|------------------
INITC     | PRODCT
          | TURBBL
          | YPLUSN

Get user-defined initial conditions for \( k \) and \( \varepsilon \).

**Input**

- CMUR
- CTHREE
- DUMMY
- MUT
- N1, N2
- VORT
- YPLUSN

- Constant \( C_\mu \) in formula for \( C_\mu \).
- Constant \( C_3 \) in formula for \( C_\mu \).
- Distance to the nearest solid wall.
- Turbulent viscosity \( \mu \) at time level \( n \).
- Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions.
- Production rate of turbulent kinetic energy.
- Nondimensional distance \( y^+ \) from the nearest solid wall.

**Output**

- E, EL
- KE, KEL
- MUTL

- Turbulent dissipation rate \( \varepsilon \) at time levels \( n \) and \( n - 1 \).
- Turbulent kinetic energy \( k \) at time levels \( n \) and \( n - 1 \).
- Turbulent viscosity \( \mu \) at time level \( n - 1 \).

**Description**

Subroutine KEINIT supplies the user-defined initial values of the turbulent kinetic energy \( k \) and the turbulent dissipation rate \( \varepsilon \). In general, this subroutine will be tailored to the problem being solved, and supplied by user. Details on the variables to be supplied by KEINIT are presented in Section 5.1 of Volume 2.

A default version of KEINIT is supplied with Proteus that computes the initial values of \( k \) and \( \varepsilon \) using the assumption of local equilibrium (dissipation equals production.) The above list of input and output Fortran variables are for the default version of KEINIT.

The steps involved in the default version of KEINIT are described below.

1. Initialize \( k \) and \( \varepsilon \) to zero.
2. Call TURBBL to compute turbulent viscosity values and to locate solid walls in the computational domain.
3. Call YPLUSN to compute \( y^+ \) and the minimum distance to the nearest solid wall.
4. Call PRODCT to compute the production rate of turbulent kinetic energy.
5. Compute \( k \) and \( \varepsilon \) using

\[
C_\mu = C_\mu \left( 1 - \varepsilon C_y y^+ \right)
\]

\[
\varepsilon = \left| \frac{P_k}{R e \rho} \right|
\]
6. Set the values of \( k, \varepsilon, \) and \( \mu, \) at time level \( n - 1 \) equal to their values at time level \( n. \)

**Remarks**

1. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the minimum distance to the nearest wall. The array is filled in subroutine YPLUSN.

2. The Fortran array VORT, from the common block TURB1, is used to store the values of the production rate of turbulent kinetic energy. The array is filled in subroutine PRODCT.

\[
k = \sqrt{\frac{\mu \varepsilon}{C_m \rho}}
\]
The MAIN program is used to manage the overall solution. The steps involved are described below.

**Preliminary Steps**

1. Call INPUT to read and print the input, and perform various initialization procedures.
2. Unless this is a restart case, call GEOM to get the computational coordinates and metric data.
3. Call INITC to get the initial flow field.
4. Call BCSET to set various boundary condition parameters and flags, and to print the input boundary condition types and values.
5. Initialize the plot file, and, if requested by the user, write the initial or restart flow field into the plot file.
6. If requested by the user, print the initial or restart flow field.

---

29 The initialization procedure depends on the type of plot file being written. See the description of subroutine PLOT.
7. Compute NTSUM, the maximum total number of marching steps to be taken, and ITEND, the corresponding final index on the time marching loop. Set the initial values of ITSEQ, the time step sequence number, and ITSWCH, the time index for switching to the next sequence, both to zero.

**Time marching loop**

8. Begin the time marching loop. The loop index IT corresponds to the known time level \( n \). Each iteration of the loop thus corresponds to a step from time level \( n \) to \( n + 1 \).

9. If at the end of a time step sequence, update ITSEQ, the time step sequence number, and ITSWCH, the time index for switching to the next sequence.

10. For the first time step, and every IDTMOD'th step thereafter, call TIMSTP to compute the new time step \( \Delta t \). For every time step update the time value \( \tau \).

11. If time-dependent boundary conditions are being used, call TBC to set the boundary condition values.

12. Call EXEC to solve the equations.

13. Call EQSTAT to compute the pressure \( p \) and temperature \( T \) from the equation of state. If either is non-positive, indicating a non-physical solution, skip forward to step 17.

14. Call FTEMP to compute the laminar viscosities \( \mu_l \) and \( \lambda_l \), the laminar thermal conductivity \( k_l \), and the specific heats \( c_p \) and \( c_v \).

15. For turbulent flow, call the appropriate subroutines to compute the effective viscosity and thermal conductivity coefficients using the turbulence model specified by the user.

16. Every ICHECK time levels, call CONV to check for convergence.

17. Call TREMAIN to find out how much CPU time remains.

18. If requested by the user, or if the calculation is converged, or if non-positive pressures or temperatures were computed, or if the job is near the CPU time limit, print the flow field at time level \( n + 1 \).

19. If requested by the user, or if the calculation is converged, or if non-positive pressures or temperatures were computed, or if the job is near the CPU time limit, write the flow field at time level \( n + 1 \) into the plot file.

20. If non-positive pressures or temperatures were computed, write an error message showing the location of the non-positive values and skip forward to step 25, ending the calculation.

21. If the calculation is converged, print a message and skip forward to step 24, ending the calculation.

22. If the job is near the CPU time limit, print a message and skip forward to step 24, ending the calculation.

23. End of time marching loop. Print a message indicating the calculation did not converge.

**Final Steps**

24. If requested by the user, call REST to write the restart file.

25. If first-order time differencing and steady boundary conditions were used, call PRTHST to print the convergence history.

**Remarks**

1. The starting index for the time marching loop is ITBEG. For a non-restart case ITBEG = 1, and thus the initial starting flow field is at time level 1. For a restart case ITBEG = \( n \), where \( n \) is the time level stored in the restart file, and thus the starting flow field is the previously computed flow field at time level \( n \).

2. The ending index for the time marching loop is ITEND = ITBEG + NTSUM - 1, where NTSUM is the total number of time steps to be taken. For a non-restart case, then, the time marches from level
1 to level 1 + NTSUM. For a restart case, the time marches from level ITBEG to level ITBEG + NTSUM.

3. The logic involving NTSUM, ITSEQ, and ITSWCH is used to implement the time step sequencing option. This allows one CFL number or time increment to be used for a specified number of steps, followed by another CFL number or time increment for another specified number of steps, etc. If this option is not used, NTSUM is simply equal to NTIME(1) and ITSEQ is always 1.

4. An error message is generated and execution is stopped if the value of ITURB does not correspond to an existing turbulence model.

5. Although the calculation will stop if $p \leq 0$, as noted above in step 20, the standard output and plot file will include the time level with the non-positive values, if that is consistent with the IPRT and IPLT input parameters in namelist IO. The restart file will not be written.

---

See Section 3.1.9 of Volume 2 for details on how to invoke the time step sequencing option.
Subroutine METS

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEOM</td>
<td>OUTPUT</td>
<td>Compute metrics of nonorthogonal grid transformation.</td>
</tr>
<tr>
<td>REST</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Input

- **DXI, DETA**: Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **IDEBUG**: Debug flags.
- **IVOUT**: Flags specifying variables to be printed.
- **NOUT**: Unit number for standard output.
- **N1, N2**: Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.
- **X, Y**: Cartesian coordinates $x$ and $y$, or cylindrical coordinates $x$ and $r$.

### Output

- **ETAX, ETAY, ETAT**: Metric coefficients $\eta_x$, $\eta_y$ (or $\eta_r$ if axisymmetric), and $\eta_r$.
- **IVOUT**: Flags specifying variables to be printed (temporarily redefined for debug output of metrics.)
- **JI**: Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.
- **XIX, XIX, XIT**: Metric coefficients $\xi_x$, $\xi_y$ (or $\xi_r$ if axisymmetric), and $\xi_r$.

### Description

Subroutine METS computes the metric coefficients and the Jacobian for the generalized nonorthogonal coordinate transformation. The metric coefficients are defined in terms of the known $(x, y)$ coordinates of the computational mesh as:

\[
\begin{align*}
\xi_x &= Jy_{\eta} \\
\xi_y &= -Jx_{\eta} \\
\eta_x &= -Jy_{\xi} \\
\eta_y &= Jx_{\xi} \\
\xi_\iota &= -x_{\xi}^\xi x_{\eta}^\eta + y_{\eta}^\eta \\
\eta_\iota &= -x_{\eta}^\eta x_{\xi}^\xi - y_{\xi}^\xi
\end{align*}
\]

where $J$ is the Jacobian of the transformation, given by

\[
J^{-1} = \left( x_{\xi}^\eta y_{\eta}^\xi - x_{\eta}^\xi y_{\xi}^\eta \right)^{-1}
\]

The derivatives of $x$ and $y$ with respect to the computational coordinates are computed numerically using the same difference formulas as used for the governing equations. At interior points the centered difference formula presented in Section 5.0 of Volume 1 is used. At boundaries three-point one-sided differencing is used. For $\xi$-derivatives at the $\xi = 0$ and $\xi = 1$ boundaries,
\[ \frac{\partial f}{\partial \xi} \approx \pm \frac{-3f_w + 4f_{w+1} - f_w + 2}{2\Delta \xi} \]

where \( w \) represents the \( \xi \)-index at the boundary (i.e., either 1 or \( N_i \)). Where a \( \pm \) sign appears, the + sign is used at the \( \xi = 0 \) boundary, and the - sign is used at the \( \xi = 1 \) boundary. An analogous formula is used for \( \eta \)-derivatives at the \( \eta = 0 \) and \( \eta = 1 \) boundaries.

**Remarks**

1. Since the current version of *Proteus* is limited to meshes that do not vary with time, the derivatives \( x \), and \( y \), are set equal to zero.
2. This subroutine generates the output for the IDEBUG(7) option.
3. An error message is generated and execution is stopped if the grid transformation Jacobian \( J \) changes sign or equals zero. This indicates that the computational mesh contains crossed or coincident grid lines. The error message is followed by a printout of the Cartesian coordinates, the Jacobian, and the metric coefficients.
Subroutine OUTPUT (LEVEL)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>PRTOUT</td>
<td>Manage printing of output.</td>
</tr>
<tr>
<td>METS</td>
<td>VORTEX</td>
<td></td>
</tr>
</tbody>
</table>

Input

- **CP, CV**: Specific heats $c_p$ and $c_v$.
- **DTAU**: Time step $\Delta t$.
- **DUMMY**: A two-dimensional scratch array dimensioned (N1P,N2P).
- **DXI, DETA**: Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **E, KE**: Turbulent dissipation rate $\varepsilon$ and kinetic energy $k$.
- **ETAX, ETAY, ETAT**: Metric coefficients $\eta_x$, $\eta_y$ (or $\eta$, if axisymmetric), and $\eta_z$.
- **GAMR**: Reference ratio of specific heats $\gamma_r$.
- **GC**: Proportionality factor $g_c$ in Newton's second law.
- **IAAXI**: Flag for axisymmetric flow.
- **ISWIRL**: Flag for swirl in axisymmetric flow.
- **IVOUT**: Flags specifying variables to be printed.
- **JI**: Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.
- **LEVEL**: Time level being printed.
- **LWALL1, LWALL2**: Flags specifying wall locations for $\xi$ and $\eta$ boundaries.
- **MACHR**: Reference Mach number $M_r$.
- **MU, LA, KT**: Effective coefficient of viscosity $\mu$, effective second coefficient of viscosity $\lambda$, and effective coefficient of thermal conductivity $k$.
- **MUT**: Turbulent viscosity coefficient $\mu_t$.
- **NOUT**: Unit number for standard output.
- **N1, N2**: Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.
- **P, T**: Static pressure $p$ and temperature $T$.
- **PR**: Reference pressure $p_r$.
- **PRR**: Reference Prandtl number $Pr_r$.
- **PRT**: Turbulent Prandtl number $Pr_t$.
- **RGAS**: Gas constant $R$.
- **RHO, U, V, W, ET**: Static density $\rho$, velocities $u$, $v$, and $w$, and total energy $E_T$.
- **RHOR, TR, UR**: Reference density $\rho_r$, temperature $T_r$, and velocity $u_r$.
- **TAU**: Time value $\tau$.
- **X, Y**: Cartesian coordinates $x$ and $y$, or cylindrical coordinates $x$ and $r$.
- **XIX, XIY, XIT**: Metric coefficients $\xi_x$, $\xi_y$ (or $\xi$, if axisymmetric), and $\xi_z$.
Output

ATITLE
   A 20-character title for variable being printed.

DUMMY
   A two-dimensional array containing the variable to be printed.

Description

Subroutine OUTPUT manages the printing of the standard output. The variables available for printing are listed and defined in Table 3-3 of Volume 2. The user-specified array IVOUT controls which variables are printed.

Each variable to be printed is stored, in turn, in the scratch array DUMMY, from the common block DUMMY1. The title printed with the variable is stored in the character array ATITLE. Subroutine PRTOUT is then called to execute the actual write statements.

Remarks

1. A warning message is printed if a non-existent output variable is requested. The printout will continue with the next requested output variable.
2. For output options 30, 31, 34, and 35, involving the pressure $p$, the value stored internally in the Proteus code is redefined as follows:

$$ p = \frac{\rho u_p^2}{p \rho g_c} $$

This is necessary because input and output values of $p$ are nondimensionalized by the reference pressure $p_r = \rho, R T_r$, while internal to the code itself $p$ is nondimensionalized by the normalizing pressure $p_n = \rho, u^2$. See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.
3. The definitions of $k_t$ and $k_p$ (IVOUT = 92 and 102) assume a constant turbulent Prandtl number is being specified in namelist TURB. If the input value of PRT $\leq 0$, indicating the use of a variable turbulent Prandtl number, the printed values of $k_t$ and $k_p$ will be incorrect.
Subroutine OUTW (LEVEL)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td></td>
<td>Compute and print parameters at boundaries.</td>
</tr>
</tbody>
</table>

**Input**

- **CP**
  - Specific heat \( c_p \).
- **DXI, DETA**
  - Computational grid spacing \( \Delta \xi \) and \( \Delta \eta \).
- **ETAX, ETAY**
  - Metric coefficients \( \eta_x \) and \( \eta_y \) (or \( \eta_x \), if axisymmetric.)
- **GC**
  - Proportionality factor \( g_c \) in Newton's second law.
- * **IWOUT1, IWOUT2**
  - Flags specifying for which boundaries parameters are to be printed.
- **LEVEL**
  - Time level being printed.
- **MU, KT**
  - Effective coefficients of viscosity \( \mu \) and thermal conductivity \( k \).
- * **NOUT**
  - Unit number for standard output.
- * **N1, N2**
  - Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions.
- **P, T**
  - Static pressure \( p \) and temperature \( T \).
- **PR**
  - Reference pressure \( p_r \).
- **PRR**
  - Reference Reynolds number \( Re_r \).
- * **RER**
  - Reference Reynolds number \( Re_r \).
- * **RHOR, UR**
  - Reference density \( \rho_r \), and velocity \( u_r \).
- **U, V, W**
  - Velocities \( u, v, \) and \( w \).
- **X, Y**
  - Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \).
- **XIX, XIY**
  - Metric coefficients \( \xi_x \) and \( \xi_y \) (or \( \xi \), if axisymmetric.)

**Output**

- None.

**Description**

Subroutine OUTW computes and prints various parameters along the computational boundaries. The variables available for printing are listed and defined in Table 3-3 of Volume 2. The user-specified arrays IWOUT1 and IWOUT2 specify at which boundaries parameters are printed, and whether normal derivatives are to be computed using two-point or three-point one-sided differencing.

The parameters printed are the Cartesian coordinates \( x \) and \( y \), the static pressure \( p \), the skin friction coefficient \( c_f \), the shear stress \( \tau_w \), the static temperature \( T \), the heat transfer coefficient \( h \), the heat flux \( q_w \), and the Stanton number \( St \). Note that some of these are meaningful only if the boundary is a solid wall.

The skin friction coefficient is defined as

\[
\frac{1}{2} \frac{\rho u^2}{\mu} = \frac{2}{Re_r} \mu \frac{\partial V_i}{\partial n}
\]

220 4.0 Proteus Subprograms: OUTW Proteus 2-D Programmer's Reference
where the overbar denotes a dimensional quantity. In this equation $\partial V_\theta / \partial n$ represents the normal derivative of the tangential velocity, with the normal vector $\vec{n}$ directed into the flow field.

For a $\zeta$ boundary, the tangential velocity for non-swirl cases is simply

$$V_t = V_\eta$$

where $V_\eta$ is the velocity in the $\eta$ direction. For axisymmetric cases with swirl, the tangential velocity on a $\zeta$ boundary is computed as

$$V_t = \sqrt{V_\eta^2 + w^2}$$

where $w$ is the swirl velocity. From the description of subroutine BCVDIR,

$$V_\eta = \frac{1}{m} (-\xi_x u + \xi_y v)$$

where

$$m = \sqrt{\xi_x^2 + \xi_y^2}$$

Using the equations in Section 6.4 of Volume 1, $\partial V_\theta / \partial n$ for a $\zeta$ boundary is thus computed as

$$\frac{\partial V_t}{\partial n} = \pm \frac{1}{m} \left[ \frac{\partial V_t}{\partial \xi} (\xi_x^2 + \xi_y^2) + \frac{\partial V_t}{\partial \eta} (\xi_x \eta_x + \xi_y \eta_y) \right]$$

where the + sign is used at the $\zeta = 0$ boundary, and the - sign is used at the $\zeta = 1$ boundary.

For an $\eta$ boundary, the tangential velocity for non-swirl cases is

$$V_t = V_\zeta$$

and for axisymmetric cases with swirl, it is computed as

$$V_t = \sqrt{V_\zeta^2 + w^2}$$

The $\zeta$ velocity is given by

$$V_\zeta = \frac{1}{m} (\eta_x u + \eta_y v)$$

where

$$m = \sqrt{\eta_x^2 + \eta_y^2}$$

Thus, for an $\eta$ boundary,

$$\frac{\partial V_t}{\partial n} = \pm \frac{1}{m} \left[ \frac{\partial V_t}{\partial \xi} (\eta_x \eta_x + \eta_y \eta_y) + \frac{\partial V_t}{\partial \eta} (\eta_x^2 + \eta_y^2) \right]$$

The shear stress $\tau_w$ is defined as

$$\tau_w = \mu \frac{\partial V_t}{\partial n}$$

$\tau_w$ is thus nondimensionalized by $\mu u / L$.

The heat flux $q_w$ is defined as
\[ q_w = -k \frac{\partial T}{\partial n} \]

where \( \partial T/\partial n \) represents the normal derivative of the temperature. For a \( \xi \) boundary,

\[
\frac{\partial T}{\partial n} = \pm \frac{1}{m} \left[ \frac{\partial T}{\partial \xi} \left( \xi_x^2 + \xi_y^2 \right) + \frac{\partial T}{\partial \eta} \left( \xi_x \eta_x + \xi_y \eta_y \right) \right]
\]

where

\[
m = \sqrt{\xi_x^2 + \xi_y^2}
\]

For an \( \eta \) boundary,

\[
\frac{\partial T}{\partial n} = \pm \frac{1}{m} \left[ \frac{\partial T}{\partial \eta} \left( \eta_x^2 + \eta_y^2 \right) + \frac{\partial T}{\partial \xi} \left( \eta_x \xi_x + \eta_y \xi_y \right) \right]
\]

where

\[
m = \sqrt{\eta_x^2 + \eta_y^2}
\]

\( q_w \) is thus nondimensionalized by \( k, T_\infty / L \).

The heat transfer coefficient \( h \) is defined as

\[
h = \frac{q_w}{T - T_r} = \frac{-k}{T - T_r} \frac{\partial T}{\partial n}
\]

This is the nondimensional form of the equation

\[
\overline{h} = \frac{\overline{q}_w}{\overline{T} - T_r} = \frac{-\overline{k}}{\overline{T} - T_r} \frac{\overline{T}}{\partial \overline{T}}
\]

\( h \) is thus nondimensionalized by \( k / L_r \).

The Stanton number \( St \) is defined as

\[
St = \frac{h}{\rho \mu \overline{C}_p} = \frac{h}{\overline{C}_p} \frac{1}{Re Pr}
\]
Subroutine PAK (IDIR,NOLD1,NOLD2)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEOM</td>
<td>CUBIC</td>
<td>Manage packing and/or interpolation of grid points.</td>
</tr>
<tr>
<td></td>
<td>ROBTS</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- **IDIR**
  Direction flag; 1 if grid points are being redistributed in the \( \xi \) direction, 2 if in the \( \eta \) direction.

- **IPACK**
  Flags for grid packing option.

- **NOLD1, NOLD2**
  Number of grid points in the \( \xi \) and \( \eta \) directions in the original grid.

- **NOUT**
  Unit number for standard output.

- **N1, N2**
  Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions in the new grid.

- **SQ**
  An array specifying the location and amount of packing.

- **X, Y**
  Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \), in the old grid.

**Output**

- **X, Y**
  Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \), in the new grid.

**Description**

Subroutine PAK manages the redistribution of the user-specified points in the computational coordinate system. It is called whenever grid packing is used. It is also called when interpolation is necessary because the computational coordinates are specified by reading them from a separate file (the NGEOM = 10 option in subroutine GEOM), and the number of points in the file is different from the number of points to be used in the calculation. PAK is called once for each direction in which points are being redistributed.

The steps involved in subroutine PAK are described below. For clarity, this discussion assumes IDIR = 1 (i.e., we are redistributing points in the \( \xi \) direction.) An exactly analogous procedure is used for IDIR = 2.

1. Set NNEW and NOLD equal to the index limits in the \( \xi \) direction for the new and old grids. Also set NOPP equal to the index limit in the \( \eta \) direction for the old grid.

2. Get \((a_\xi)\_i\) the normalized physical arc length along a coordinate line in the \( \xi \) direction, from the beginning of the line to each grid point in the new grid. The normalizing distance is the total arc length of the line, and thus these arc lengths apply to any coordinate line in the \( \xi \) direction. If the points are not being packed in the \( \xi \) direction, but only interpolated, then

\[
(a_\xi)_i = \frac{i - 1}{NNEW - 1}
\]

for \( i = 1 \) to \( NNEW \). In the new grid, the points will thus be evenly distributed in physical space along each coordinate line in the \( \xi \) direction. If the grid points are being packed in the \( \xi \) direction, subroutine ROBTS is called to compute \((a_\xi)\_i\) from the packing parameters specified by the user.

3. Begin loop from IOPP = 1 to NOPP. This loop thus runs over the points in the \( \eta \) direction in the old grid. We will be redistributing points in the \( \xi \) direction for each \( \eta \) value in the old grid.
4. Get \((a_{UP})_i\), the normalized physical arc length along a coordinate line in the \(\xi\) direction, from the beginning of the line to each grid point in the old grid. These values are found by first computing the non-normalized arc lengths, as follows:

\[
(a_{UP})_1 = 0
\]

\[
(a_{UP})_i = (a_{UP})_{i-1} + \sqrt{(x_{i,j} - x_{i-1,j})^2 + (y_{i,j} - y_{i-1,j})^2}
\]

for \(i = 2\) to \(\text{NOLD1}\). These values are normalized by setting

\[
(a_{UP})_i = \frac{(a_{UP})_i}{(a_{UP})_{\text{NOLD1}}}
\]

for \(i = 1\) to \(\text{NOLD1}\). To eliminate any problems with roundoff error, \((a_{UP})_{\text{NOLD1}}\) is explicitly set equal to 1.

5. Given \(x\) and \(a_{UP}\) for the old grid, and \(a_P\) for the new grid, call CUBIC to interpolate for \(x\) in the new grid. Similarly interpolate for \(y\).

6. Redefine the Fortran variables \(X\) and \(Y\) as the \(x\) and \(y\) coordinates in the new grid.

7. End of loop over the points in the \(\eta\) direction in the old grid.

Remarks

1. In the Fortran code, the comments sometimes refer to the “packing” direction. This terminology actually means the direction in which grid points are being redistributed, even if they are not being packed but only interpolated. Similarly, references to the “packed” and “unpacked” grid actually mean the new and old grids.

2. An error message is generated and execution is stopped if an invalid grid packing option is requested.
Subroutine PERIOD

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td></td>
<td>Define extra line of data for use in computing coefficients for spatially periodic boundary conditions.</td>
</tr>
<tr>
<td>EXECT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- **CP, CV**
  Specific heats $c_p$ and $c_v$ at time level $n$.

- **E, EL**
  Turbulent dissipation rate $\epsilon$ at time levels $n$ and $n-1$.

- **ETAX, ETAY, ETAT**
  Metric coefficients $\eta_x, \eta_y$ (or $\eta_r$ if axisymmetric), and $\eta_r$.

- **JI**
  Inverse Jacobian of the nonorthogonal grid transformation, $J^{-1}$.

- **KBCPER**
  Flags for spatially periodic boundary conditions in the $\xi$ and $\eta$ directions; 0 for non-periodic, 1 for periodic.

- **KE, KEL**
  Turbulent kinetic energy $k$ at time levels $n$ and $n-1$.

- **MU, LA, KT**
  Effective coefficient of viscosity $\mu$, effective second coefficient of viscosity $\lambda$, and effective coefficient of thermal conductivity $k$.

- **MUT, MUTL**
  Turbulent viscosity $\mu_t$ at time levels $n$ and $n-1$.

- **NPT1, NPT2**
  $N_1$ and $N_2$ for non-periodic boundary conditions, $N_1 + 1$ and $N_2 + 1$ for spatially periodic boundary conditions in $\xi$ and $\eta$.

- **N1, N2**
  Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.

- **P, T**
  Static pressure $p$ and temperature $T$ at time level $n$.

- **RHO, U, V, W, ET**
  Static density $\rho$, velocities $u, v$, and $w$, and total energy $E_T$ at time level $n$.

- **RHOL, UL, VL, WL, ETL**
  Static density $\rho$, velocities $u, v$, and $w$, and total energy $E_T$ from previous ADI sweep.

- **TL**
  Static temperature $T$ from previous ADI sweep.

- **XIX, XIY, XIT**
  Metric coefficients $\xi_x, \xi_y$ (or $\xi_r$ if axisymmetric), and $\xi_r$.

- **Y**
  Radial coordinate $r$ for axisymmetric flow.

**Output**

All of the flow and metric-related input parameters listed above, at $i = N_1 + 1$ for periodic boundary conditions in the $\xi$ direction, and at $j = N_2 + 1$ for periodic boundary conditions in the $\eta$ direction.

**Description**

Subroutine PERIOD adds, in effect, an additional set of points at $i = N_1 + 1$ for periodic boundary conditions in the $\xi$ direction, and at $j = N_2 + 1$ for periodic boundary conditions in the $\eta$ direction. This allows us to use central differencing in the periodic direction, at $i = N_1$ and/or $j = N_2$, computing the coefficient submatrices and source term subvector in the same way as at the interior points.\(^{31}\)

For periodic boundary conditions in the $\xi$ direction, the extra points are added by setting

---

\(^{31}\) See Section 7.2.2 of Volume I for details on the solution procedure for spatially periodic boundary conditions.
\[ f_{N_1 + 1, j} = f_{2, j} \]

where \( j = 1 \) to \( N_2 \), and \( f \) represents one of the flow variables or metrics. Similarly, extra points are added at \((i, N_2 + 1)\) for periodic boundary conditions in the \( \eta \) direction.
Subroutine PLOT (LEVEL)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td></td>
<td>Write files for post-processing by CONTOUR or PLOT3D plotting programs.</td>
</tr>
</tbody>
</table>

**Input**

- CP, CV
- ETAX, ETAY
- * GAMR
- GC
- * IPLOT
- LEVEL
- * LR, UR, RHOR, TR
- * MACHR
- * NOUT
- * NPLCT
- NPLCTX
- * NSCR1
- * N1, N2
- P, T
- PR
- * RER
- * RG
- RGAS
- RHO, U, V, W, ET
- TAU
- * TITLE
- X, Y
- XIX, XIY

Specific heats $c_p$ and $c_v$.

Metric coefficients $\eta_x$ and $\eta_y$ (or $\eta$, if axisymmetric).

Reference ratio of specific heats, $\gamma_r$.

Proportionality factor $g_c$ in Newton’s second law.

Flag specifying type of plot file to be written.

Time level to be written into the file (0 for initialization, and -1 to read the scratch file and write XYZ and Q files with the IPLOT = -3 option).

Reference length $L_r$, velocity $u_r$, density $\rho_r$, and temperature $T_r$.

Reference Mach number $M_r$.

Unit number for standard output.

Unit number for writing CONTOUR file, or PLOT3D Q file.

Unit number for writing PLOT3D XYZ file.

Unit number for scratch file.

Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.

Static pressure $p$ and temperature $T$.

Reference pressure $p_r$.

Reference Reynolds number $Re_r$.

Dimensional gas constant $\bar{R}$.

Dimensionless gas constant $R$.

Static density $\rho$, velocities $u$, $v$, and $w$, and total energy $E_T$.

Current time value $\tau$.

Case title.

Cartesian coordinates $x$ and $y$, or cylindrical coordinates $x$ and $r$.

Metric coefficients $\xi$, and $\eta$, (or $\xi$, if axisymmetric).

**Output**

None.

**Description**

Subroutine PLOT writes a file or files, commonly called plot files, for post-processing by the CONTOUR or PLOT3D plotting programs. The type of files written is controlled by the user-specified parameter IPLOT. The format and contents of the different types of plot files are described in detail in Section 4.2 of Volume 2. They are therefore described only briefly here.
CONTOUR Plot File \((\text{IPLOT} = \pm 1)\)

If \(\text{IPLOT} = 1\), a CONTOUR plot file is written with the title and reference conditions included at each time level. The value of \(n\) is written into the header for each time level, but \(\tau\), the time itself, is not written into the file. No initialization step is necessary.

If \(\text{IPLOT} = -1\), a CONTOUR plot file is also written, but the title and reference conditions are written only at the beginning of the file. In addition the time \(\tau_{n,j}\) is written into the file at each time level. In this case the initialization step consists of writing the title and reference conditions at the beginning of the file.

PLOT3D/WHOLE Plot Files \((\text{IPLOT} = 2)\)

If \(\text{IPLOT} = 2\), XYZ and Q files are written in PLOT3D/WHOLE format. The XYZ file is written only during the initialization step. The Q file is written at each time level requested by the user. The Q file will thus consist of multiple sets of data, each containing the computed results at a single time level. The time \(\tau_{1,1}\) is written into the header for each set of data in the Q file. Since Proteus 2-D is two-dimensional, \(N_3\), the number of points in the \(z\) direction in the XYZ and Q files, is set equal to 1.

PLOT3D/PLANES Plot Files \((\text{IPLOT} = 3)\)

If \(\text{IPLOT} = 3\), XYZ and Q files are written in PLOT3D/PLANES format. Since Proteus 2-D is two-dimensional, \(N_3\), the number of points in the \(z\) direction in the XYZ and Q files, is set equal to 1. This makes the XYZ and Q files identical to those created using the \(\text{IPLOT} = 2\) option.

PLOT3D/PLANES Plot Files \((\text{IPLOT} = -3)\)

The files created with this option are similar to those created with the \(\text{IPLOT} = 3\) option, except the time \(\tau_{n,j}\) is written into the \(z\) slot in the XYZ file, and the number of points in the \(z\) direction is set equal to the number of time levels in the XYZ and Q files.

However, because the calculation may converge or become non-physical, the number of time levels that end up being written into the files is not known until the end of the Proteus run. Therefore, as the calculation proceeds the results are actually written into a scratch file. \(N_3\), the counter for the number of time levels, is set equal to zero in the initialization step and updated each time a time level is added to the scratch file. At the end of the Proteus run the scratch file is read and the XYZ and Q files are written.

PLOT2D Plot Files \((\text{IPLOT} = 4)\)

If \(\text{IPLOT} = 4\), XYZ and Q files are written in PLOT3D's 2D format. The XYZ file is written only during the initialization step. The Q file is written at each time level requested by the user. The Q file will thus consist of multiple sets of data, each containing the computed results at a single time level. The time \(\tau_{1,1}\) is written into the header for each set of data in the Q file.

Remarks

1. For the CONTOUR plot file, the \(\text{IPLOT} = -1\) option is the better one to use. The \(\text{IPLOT} = 1\) option is included only to be consistent with the various PLOT3D options.
2. In defining the pressure to be written into the CONTOUR plot file, the value stored internally in the Proteus code is redefined as follows:

   \[
   p = p - \frac{p \rho R T_0}{\rho \delta e}
   \]

   This is necessary because input and output values of \(p\) are nondimensionalized by the reference pressure \(p = \rho R T_0\), while internal to the code itself \(p\) is nondimensionalized by the normalizing pressure \(p_e = \rho \delta d\). See Section 3.1.1 of Volume 2 for a discussion of the distinction between reference and normalizing conditions.
3. The current version of PLOT3D does not work for multiple time levels, although future versions might. Thus the IPLOT = 2, 3, and 4 options, while containing multiple time levels, cannot easily be used to create plots showing the time development of the flow. You can, however, fake it out using the IPLOT = -3 option. With this option, plots can be generated at different time levels by plotting at different PLOT3D "z" stations.

4. Note that the time \( \tau_{1,1} \) written into the Q file header with the IPLOT = 2, 3, and 4 options is the time at the point \( \xi = \eta = 0 \). If the input variable IDTAU = 5 or 6, \( \tau \) will vary in space and therefore \( \tau_{i,j} \neq \tau_{1,1} \).

5. To save storage, the common variable AMATI, which is normally used for the subdiagonal submatrix of coefficients in the block tridiagonal system of equations, is equivalenced to the local three-dimensional variable Q used to store the Q variables that are written into the PLOT3D Q file.

6. PLOT3D assumes that velocity is nondimensionalized by the reference speed of sound \( a = (\gamma R T) \), and that energy is nondimensionalized by \( \rho a^2 \). In Proteus these variables are nondimensionalized by \( u \) and \( \rho u^2 \). That is why the reference Mach number \( M_r \) appears in the definitions of Q(1,2) through Q(1,5).

7. An error message is generated and execution is stopped if an illegal plot file option is requested.
Subroutine PRODCT

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEINIT</td>
<td></td>
<td>Compute production term for the k-ε turbulence model.</td>
</tr>
<tr>
<td>TURBCH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input**

- **DXI, DETA**: Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- **ETAX, ETAY**: Metric coefficients $\eta_x$ and $\eta_y$ (or $\eta$ if axisymmetric).
- **IAXI**: Flag for axisymmetric flow.
- **MUT**: Turbulent viscosity $\mu_t$ at time level $n$.
- **N1, N2**: Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.
- **RER**: Reference Reynolds number $Re_r$.
- **U, V, W**: Velocities $u$, $v$, and $w$ at time level $n$.
- **XIX, XIY**: Metric coefficients $\xi_x$, and $\xi_y$ (or $\xi$, if axisymmetric).
- **Y**: Radial coordinate $r$ for axisymmetric flow.

**Output**

- **PONE, PTWO**: Parts 1 and 2 of the production rate of turbulent kinetic energy.
- **VORT**: Production rate of turbulent kinetic energy.

**Description**

Subroutine PRODCT computes the turbulent kinetic energy production rate using

$$ P_k = \frac{\mu_t}{Re_r} P_1 - \frac{2}{3} \rho k P_2 $$

where, for 2-D planar flow,

$$ P_1 = 2 \left[ \left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2 \right] - \frac{2}{3} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right)^2 + \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 $$

$$ P_2 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} $$

and for axisymmetric flow,

$$ P_1 = 2 \left[ \left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial r} \right)^2 + \left( \frac{\nu}{r} \right)^2 \right] - \frac{2}{3} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} + \frac{\nu}{r} \right)^2 + \left( \frac{\partial u}{\partial r} + \frac{\partial v}{\partial x} \right)^2 $$

$$ + \left( \frac{\partial w}{\partial x} \right)^2 + \left[ r \frac{\partial}{\partial r} \left( \frac{w}{r} \right) \right]^2 $$

$$ P_2 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} + \frac{\nu}{r} $$
To evaluate the spatial derivatives, the centered difference formulas presented in Section 5.0 of Volume 1 are used at interior points, and second-order one-sided difference formulas are used at boundary points.

Remarks

1. To save storage space, this subroutine uses the Fortran variable VORT to store the turbulent kinetic energy production rate. Care must be taken when this subroutine is used together with subroutine VORTEX.
Subroutine PRTHST

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td></td>
<td>Print convergence history.</td>
</tr>
</tbody>
</table>

**Input**

- **ICHECK**  
  Convergence checking interval.
- **IREST**  
  Flag for restart file; 0 for no restart file, 1 to write a restart file, 2 to read and write a restart file.
- **IT**  
  Last computed time step number n.
- **ITBEG**  
  The time level n at the beginning of a run.
- **NC, NXM, NYM, NZM, NEN**  
  Array indices associated with the continuity, x-momentum, y-momentum (or r-momentum if axisymmetric), swirl momentum, and energy equations.
- **NEQ**  
  Number of coupled equations being solved, Nₑₑₑₑ.
- **NHIST**  
  Unit number for convergence history file.
- **NHMAX**  
  Maximum number of time levels allowed in the printout of the convergence history file (not counting the first two, which are always printed.)
- **NOUT**  
  Unit number for standard output.

**Output**

None.

**Description**

Subroutine PRTHST prints the convergence history as part of the standard output. The information is obtained from the unformatted convergence history file written in subroutine RESID. The parameters printed are described in Section 4.1.6 of Volume 2, and the unformatted convergence history file is described in Section 4.3 of Volume 2. To avoid undesirably long tables, the convergence parameters are printed at an interval that limits the printout to NHMAX time levels. As described in Section 4.1.6 of Volume 2, however, they are always printed at the first two time levels.
Subroutine PRTOUT (ATITLE, LEVEL, AVAR)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td></td>
<td>Print output.</td>
</tr>
</tbody>
</table>

**Input**

- **ATITLE**: A 20-character title for variable being printed.
- **AVAR**: A two-dimensional array containing the variable to be printed.
- **DTAU**: Time step Δt.
- **IDTAU**: Flag for time step selection method.
- **IPRT1A, IPRT2A**: Indices for printout in the ξ and η directions.
- **LEVEL**: Time level to be printed.
- **LR, UR**: Reference length L, and velocity u.
- **NOUT**: Unit number for standard output.
- **NPRT1, NPRT2**: Total number of indices for printout in the ξ and η directions.
- **TAU**: Current time value t.

**Output**

None.

**Description**

Subroutine PRTOUT performs the actual printing of the standard output file. It prints the variable AVAR, with the title ATITLE. The output is printed in columns running in the η direction. The rows run in the ξ direction. If the results at every grid point are printed, there will be a total of \(N_2\) columns, each with \(N_1\) rows. The columns are grouped in super-rows of up to 10 columns each.

The steps involved are as follows:

1. Set the total number of columns, and rows per super-row.
2. Redefine AVAR, the input array containing the variable to be printed, including only the elements requested.
3. Determine the number of super-rows. If NCOL is not exactly divisible by 10, the last super-row will have less than 10 columns.
4. Print the title for the variable. If the time step is constant in space, the dimensional time \(t\) and time step \(Δt\) are printed with the title.
5. Begin loop over the number of super-rows.
6. Set NC1 and NC2 equal to the number of the first and last column in this super-row. (I.e., for the first super-row NC1 and NC2 will be 1 and 10, for the second they will be 11 and 20, etc. For the last super-row, NC2 will be NCOL.)
7. Print the heading for the super-row, labeling each column with the proper ξ index.
8. Print the super-row itself, labeling each row with the proper η index.
9. End of loop over the number of super-rows.
Subroutine RESID (IAVR,S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>ISAMAX, SASUM, SNRM2</td>
<td>Compute residuals and write convergence history file.</td>
</tr>
</tbody>
</table>

**Input**

- **CHGAVG**: Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps, $\Delta Q_{av}$.
- **CHGMAX**: Maximum change in absolute value of the dependent variables over previous time step (or NITAVG - 1 time steps if ICTEST = 2), $\Delta Q_{max}$.
- **DTAU**: Time step $\Delta t$.
- **DUMMY**: A two-dimensional scratch array.
- **EPS**: Convergence level to be reached, $\varepsilon$.
- **IAVR**: Flag specifying whether residual is computed without or with the artificial viscosity terms; 1 for without, 2 for with.
- **IAV2E, IAV4E**: Flags for second- and fourth-order explicit artificial viscosity.
- **ICHECK**: Convergence checking interval.
- **ICTEST**: Flag for convergence criteria to be used.
- **IDTAU**: Flag for time step selection method.
- **IHSTAG**: Flag for constant stagnation enthalpy option.
- **ISWIRL**: Flag for swirl in axisymmetric flow.
- **IT**: Current time step number $n$.
- **ITBEG**: The time level $n$ at the beginning of a run.
- **LR, UR**: Reference length $L$, and velocity $u$.
- **NEQ**: Number of coupled equations being solved, $N_q$.
- **NHIST**: Unit number for convergence history file.
- **NITAVG**: Number of time steps in moving average convergence test.
- **NPT1, NPT2**: $N_1$ and $N_2$ for non-periodic boundary conditions, $N_1 + 1$ and $N_2 + 1$ for spatially periodic boundary conditions in $\xi$ and $\eta$.
- **NTOTP**: Dimensioning parameter specifying the storage required for a full two-dimensional array (i.e., $N1P \times N2P$).
- **NVD, NPTSD**: Leading two dimensions for the array $S$.
- **N1, N2**: Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.
- **N1P, N2P**: Parameters specifying the dimension sizes in the $\xi$ and $\eta$ directions.
- **S**: Source term subvector $S$ for first ADI sweep.
- **TAU**: Current time value $\tau$.

---

Proteus 2-D Programmer's Reference
Output

LRMAX
RESAVG
RESL2
RESMAX

Grid indices $i$ and $j$, in the $\xi$ and $\eta$ directions, corresponding to the location of RESMAX.
The average absolute value of the residual, $R_{av}$.
The $L_2$ norm of the residual, $R_{L2}$.
The maximum absolute value of the residual, $R_{max}$.

Description

Subroutine RESID computes various measures of the residual, and writes the convergence history file.

For problems without artificial viscosity, the steady-state form of the governing partial differential equations can be written as

$$0 = -\frac{\partial \hat{E}}{\partial \xi} - \frac{\partial \hat{F}}{\partial \eta} + \frac{\partial \hat{E}_\nu}{\partial \xi} + \frac{\partial \hat{F}_\nu}{\partial \eta}$$

The residual is defined as the number resulting from evaluating the right hand side of the above equation. For first-order time differencing, this is simply the source term for the first ADI sweep, divided by the time step $\Delta t$. The residual at a specific grid point and time level is thus

$$R_{i,j}^n = \frac{S_{i,j}}{(\Delta t)^n}$$

where $S$ is the source term for the first ADI sweep. Separate residuals are computed for each governing equation.

Adding artificial viscosity, however, changes the governing equations. With artificial viscosity, the difference equations actually correspond to the following differential equations at steady state.

$$0 = -\frac{\partial \hat{E}}{\partial \xi} - \frac{\partial \hat{F}}{\partial \eta} + \frac{\partial \hat{E}_\nu}{\partial \xi} + \frac{\partial \hat{F}_\nu}{\partial \eta}$$

$$+ \frac{\theta_2}{f} \left[ (\Delta \xi)^2 \frac{\partial^2 (\hat{Q})}{\partial \xi^2} + (\Delta \eta)^2 \frac{\partial^2 (\hat{Q})}{\partial \eta^2} \right]$$

$$- \frac{\theta_3}{f} \left[ (\Delta \xi)^4 \frac{\partial^4 (\hat{Q})}{\partial \xi^4} + (\Delta \eta)^4 \frac{\partial^4 (\hat{Q})}{\partial \eta^4} \right]$$

For cases run with artificial viscosity, therefore, the residual should include the explicit artificial viscosity terms. The implicit terms do not appear, since they difference $\Delta \hat{Q}$, and in the steady form of the equations $\Delta \hat{Q} = 0$. Since the explicit artificial viscosity terms are added to the source term for the first ADI sweep, they are automatically included in the residual.

Three measures of the residual are computed for each governing equation - the $L_2$ norm of the residual, the average absolute value of the residual, and the maximum absolute value of the residual. In addition, the $(\xi, \eta)$ indices corresponding to the location of the maximum residual are saved. The $L_2$ norm of the residual is defined as

---

32 See equation (8.5a) in Volume 1. For first-order time differencing, $\theta_2 = \theta_3 = 0$.

33 These equations represent the use of the constant coefficient artificial viscosity model. The nonlinear coefficient model is more complicated, but the same principle applies.
In computing the residuals, the summations, maximums, and averages are over all interior grid points, plus points on spatially periodic boundaries.

For cases run with artificial viscosity, subroutine RESID is called from EXEC both before and after the artificial viscosity terms have been added to the equations. The residuals are thus computed both with and without the artificial viscosity terms. This may provide some estimate of the overall error in the solution introduced by the artificial viscosity. Convergence is determined by the residuals with the artificial viscosity terms included.

In addition to computing the residuals, subroutine RESID writes the convergence history file. The contents and format of this file are described in detail in Section 4.3 of Volume 2.

Remarks

1. The Cray BLAS routines SNRM2 and SASUM are used in computing the $L_2$ norm of the residual and the average absolute value of the residual, respectively. The Cray search routine ISAMAX is used in computing the maximum absolute value of the residual.

2. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the residual at each grid point.
Subroutine REST (IOPT)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITC</td>
<td>METS</td>
<td>Read and/or write restart file.</td>
</tr>
<tr>
<td>MAIN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input When Reading the Restart File

- **GAMR**: Reference ratio of specific heats, \( \gamma \).
- **HSTAG**: Stagnation enthalpy \( h_T \) used with constant stagnation enthalpy option.
- **IHSTAG**: Flag for constant stagnation enthalpy option.
- **IOPT**: Flag specifying I/O operation; 1 to read, 2 to write.
- **ITURB**: Flag for turbulent flow option.
- **NRQIN**: Unit number for reading the restart flow field.
- **NRXIN**: Unit number for reading the restart computational mesh.
- **RGAS**: Dimensionless gas constant \( R \).

Input When Writing the Restart File

- **E, KE**: Turbulent dissipation rate \( \varepsilon \) and kinetic energy \( k \) at time level \( n + 1 \).
- **EL, KEL**: Turbulent dissipation rate \( \varepsilon \) and kinetic energy \( k \) at time level \( n \).
- **IOPT**: Flag specifying I/O operation; 1 to read, 2 to write.
- **IT**: Current time step number \( n \).
- **ITURB**: Flag for turbulent flow option.
- **MACHR**: Reference Mach number \( M_r \).
- **NRQOUT**: Unit number for writing the restart flow field.
- **NRXOUT**: Unit number for writing the restart computational mesh.
- **N1, N2**: Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions.
- **RER**: Reference Reynolds number \( R_e \).
- **RHO, U, V, W, ET**: Static density \( \rho \), velocities \( u, v, \) and \( w \), and total energy \( E_T \) at time level \( n + 1 \).
- **RHOL, UL, VL, WL, ETL**: Static density \( \rho \), velocities \( u, v, \) and \( w \), and total energy \( E_T \) at time level \( n \).
- **TAU**: Computational time \( \tau \) at time level \( n + 1 \).
- **X, Y**: Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \).

Output When Reading the Restart File

- **DXI, DETA**: Computational grid spacing \( \Delta \xi \) and \( \Delta \eta \).
- **E, KE**: Turbulent dissipation rate \( \varepsilon \) and kinetic energy \( k \) at time level ITBEG.
Output When Writing the Restart File

None.

Description

Subroutine REST reads and/or writes the restart files. Restarting a calculation requires two unformatted files - one containing the computational mesh and one containing the flow field.

If subroutine REST is being used to read the restart files, the computational mesh is first read from unit NRXIN. The grid increments $\Delta_x$ and $\Delta_y$ are then set, and subroutine METS is called to compute the metric coefficients and the Jacobian of the grid transformation.

The flow field file is read next, from unit NRQIN. It normally contains the results at the last two time levels that were computed during the previous run. If only one level is present in the file, however, the results at level $n - 1$ are set equal to those at level $n$. If the previous run used the two-equation turbulence model, the turbulence variables are also read from the file. The beginning time level for the time marching loop is set equal to the level stored in the restart file. The flow field variables in the restart file are the conservation variables $Q$, nondimensionalized as in the plotting program PLOT3D. They therefore must be converted into the primitive variables used in Proteus. The temperature is then computed from the perfect gas equation of state, with $c_p$ and $c_v$ defined using the input reference conditions.

When writing the restart files, the file containing the computational mesh is written onto unit NRXOUT. The primitive flow variables are then redefined as conservation variables and nondimensionalized as in PLOT3D. They are then written onto unit NRQOUT. If the current run used the two-equation turbulence model, the turbulence variables are also written into the file.

Remarks

1. If, in the input namelist RSTRT, NRXOUT and NRQOUT are set equal to NRXIN and NRQIN, respectively, the output restart files will overwrite the input restart files.

2. Except for the turbulence variables and the variables at time level $n - 1$, the restart files have the same format as the XYZ and Q files created using the I=2 and 3 options. These restart files can thus also be used as XYZ and Q files for the PLOT3D plotting program. Since $N3 = 1$, the $n - 1$ level will not be read by PLOT3D.

34 See Sections 4.2.3 and 4.4 of Volume 2.
3. The temperature $T$ is computed using the equation of state, which contains a specific heat coefficient (either $c_p$ or $c_v$, depending on whether the stagnation enthalpy is assumed constant or not.) In subroutine REST, a constant value of specific heat is used, consistent with the reference temperature $T_r$. If the user specified constant specific heat (i.e., a value for $\gamma$, was read in), this is not a problem. However, if the temperature-dependent specific heat option is being used (i.e., a value for $\gamma$, was not read in), the equation of state and the empirical equation for specific heat are coupled. For this reason, in INITC (the routine that calls REST), $T$ is recomputed by calling EQSTAT after the specific heats have been computed in FTEMP. Ideally, this coupling would be handled by iteration between FTEMP and EQSTAT. This is not currently done in Proteus, however.
Subroutine ROBTS (NP, A, B, XP)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAK</td>
<td></td>
<td>Pack points along a line using Roberts transformation.</td>
</tr>
</tbody>
</table>

**Input**

- **A**
  - Parameter $c_\text{in}$ in Roberts transformation formula specifying location of packing: 0.0 to pack near $XP = 1$ only, 1.0 to pack near $XP = 0$ only, and 0.5 to pack equally at $XP = 0$ and 1.0.

- **B**
  - Parameter $\beta$ in Roberts transformation formula specifying amount of packing. A value approaching 1.0 from above gives denser packing.

- **NP**
  - Number of grid points along the line.

**Output**

- **XP**
  - Coordinates of packed grid points along the line.

**Description**

Subroutine ROBTS packs points along a line of length one using a transformation due to Roberts (1971). The basic transformation is given by

$$xp = \frac{(\beta + 2a)\beta_{r}^{x} - \beta + 2a}{(2a + 1)(1 + \beta_{r}^{x})}$$

where

$$\beta_r = \frac{\beta + 1}{\beta - 1}$$

and $x_p$ and $x_{up}$ are the packed and unpacked (i.e., evenly spaced) coordinates along the line. The parameter $a$ determines the packing location. For $a = 0$, the points will be packed only near $x_p = 1$, and for $a = 1/2$ the points will be packed equally near $x_p = 0$ and $x_p = 1$. The packing parameter $\beta$ determines the amount of packing. It is a number greater than 1, but generally 1.1 or below. The closer $\beta$ is to 1, the tighter the packing will be.

It may seem logical to set $a = 1$ to pack points near $x_p = 0$. With the basic transformation, however, this doesn’t work. In Proteus we get around this problem by replacing $a$ in the above transformation with $a_w$, where $a_w = a$ if $a = 0$ or 1/2, and $a_w = 0$ if $a = 1$. If $a = 0$ or 1/2, no further action is necessary. If $a = 1$, however, we must invert the resulting $x_p$ values and re-order the indices. I.e., for $i = 1$ to NP, we set

$$(x_p)_i = 1 - (x_p)_i$$

After this operation, the array $x_p$ will run from 1 to 0, packed near 1. To re-order the indices, for $i = 1$ to NP we set

$$(x_p)_{NP - i + 1} = (x_p)_i$$

After this operation, $x_p$ will run from 0 to 1, packed near 0.
Finally, to ensure round-off error doesn't affect the endpoint values, we set $(x_p)_1 = 0$ and $(x_p)_{N_P} = 1$.

Remarks

1. The namelist input variable SQ(IDIR,1), which is used to specify the packing location in direction IDIR, is actually equal to $1 - a$. Therefore, setting SQ(IDIR,1) = 0 results in packing near the $\xi$ or $\eta = 0$ boundary, and SQ(IDIR,1) = 1 results in packing near the $\xi$ or $\eta = 1$ boundary.
## Function `SASUM (N,V,INC)`

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESID</td>
<td></td>
<td>Compute the sum of the absolute values of the elements of a vector.</td>
</tr>
</tbody>
</table>

### Input

- **N**
  - Number of elements in the vector to be summed.
- **V**
  - Vector to be summed.
- **INC**
  - Skip distance between elements of V. For contiguous elements, INC = 1.

### Output

- **SASUM**
  - Sum of the absolute values of the elements of V.

### Description

Function `SASUM` computes the sum of the absolute values of the elements of a vector. For a one-dimensional vector, the use of `SASUM` is straightforward. For example,

\[ sasum(np, v, 1) = \sum_{i=1}^{np} V_i \]

A starting location can be specified, as in

\[ sasum(np-4, v(5), 1) = \sum_{i=5}^{np} V_i \]

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if A is an array dimensioned NDIM1 by NDIM2, then

\[ sasum(ndim1 \times ndim2, a, 1) = \sum_{i=1}^{ndim1} \sum_{j=1}^{ndim2} A_{i,j} \]

One dimension at a time can also be summed. For example,

\[ sasum(ndim1, a(1,5), 1) = \sum_{i=1}^{ndim1} A_{i,5} \]

Similarly, by specifying a skip increment,

\[ sasum(ndim2, a(5,1), ndim1) = \sum_{j=1}^{ndim2} A_{5,j} \]

### Remarks

1. `SASUM` is a Cray BLAS (Basic Linear Algebra Subprograms) routine (Cray Research, Inc., 1989b).
Subroutine SGEFA (A,LDA,N,IPVT,INFO)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCELIM</td>
<td>ISAMAX</td>
<td>Factor a matrix using Gaussian elimination.</td>
</tr>
<tr>
<td>BVUP</td>
<td>ISAMAX</td>
<td></td>
</tr>
</tbody>
</table>

Input

A           | An array containing the matrix A to be factored, dimensioned as A(LDA,N).
LDA         | The leading dimension of the array A.
N           | The order of the matrix A.

Output

A           | An upper triangular matrix and the multipliers which were used to obtain it. The factorization can be written as A = LU, where L is a product of permutation and unit lower triangular matrices, and U is upper triangular.
IPVT        | A vector of length N containing pivot indices.
INFO        | An error flag: 0 for normal operation, k if $U_{kk}$ = 0.

Description

Subroutine SGEFA is used in combination with subroutine SGESL to solve the matrix equation $Ax = B$. If the Fortran arrays A and B represent A and B, where A is a square N by N matrix and B is a matrix (or vector) with NCOL columns, and if the leading dimension of the Fortran array A is LDA, then the Fortran sequence

```fortran
    call sgefa (a, lda, n, ipvt, info)
    do 10 j = 1, ncol
      call sgesc (a, lda, n, ipvt, b(1, j), 0)
      continue
```

computes $A^{-1}B$, storing the result in B.

Remarks

1. SGEFA is a Cray LINPACK routine (Cray Research, Inc., 1989b; Dongarra, Moler, Bunch, and Stewart, 1979).
Subroutine SGESL (A,LDA,N,IPVT,B,JOB)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCELIM</td>
<td></td>
<td>Solve the matrix equation $Ax = B$ or $A^Tx = B$ using the factors com-</td>
</tr>
<tr>
<td>BVUP</td>
<td></td>
<td>puted by SGEFA.</td>
</tr>
</tbody>
</table>

**Input**

- **A**: The two-dimensional output array A from SGEFA containing the factorization of matrix A.
- **B**: The right-hand side vector B.
- **IPVT**: The output array IPVT of pivot indices from SGEFA.
- **JOB**: Flag specifying type of matrix equation: 0 to solve $Ax = B$; non-zero to solve $A^Tx = B$.
- **LDA**: The leading dimension of the array A.
- **N**: The order of the matrix A.

**Output**

- **B**: The solution vector x.

**Description**

Subroutine SGESL is used in combination with subroutine SGEFA to solve the matrix equation $Ax = B$. See the description of subroutine SGEFA for details.

**Remarks**

1. SGESL is a Cray LINPACK routine (Cray Research, Inc., 1989b; Dongarra, Moler, Bunch, and Stewart, 1979).
Function SNRM2 (N,V,INC)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESID</td>
<td></td>
<td>Compute the $L_2$ norm of a vector.</td>
</tr>
</tbody>
</table>

**Input**

- **N**: The number of elements in the vector $V$.
- **V**: The vector whose norm is to be computed.
- **INC**: Skip distance between elements of $V$. For contiguous elements, INC = 1.

**Output**

- **SNRM2**: The $L_2$ norm of the vector $V$.

**Description**

Function SNRM2 computes the $L_2$ norm of a vector. For a one-dimensional vector, the use of SNRM2 is straightforward. For example,

$$snrm2(n, v, 1) = \left( \sum_{i=1}^{n} v_i^2 \right)^{1/2}$$

A starting location can be specified, as in

$$snrm2(n-4, v(5), 1) = \left( \sum_{i=5}^{n-4} v_i^2 \right)^{1/2}$$

Multi-dimensional arrays can be used by taking advantage of the way Fortran arrays are stored in memory, and specifying the proper vector length and skip distance. For instance, if $A$ is an array dimensioned NDIM1 by NDIM2, then

$$snrm2(ndim1*ndim2, a, 1) = \left( \sum_{i=1}^{ndim1} \sum_{j=1}^{ndim2} a_{i,j}^2 \right)^{1/2}$$

One dimension at a time can also be summed. For example,

$$snrm2(ndim1, a(1,5), 1) = \left( \sum_{i=1}^{ndim1} a_{i,5}^2 \right)^{1/2}$$

Similarly, by specifying a skip increment,
snrm2(ndim2,a(5,1),ndim1) = \left( \sum_{j=1}^{\text{ndim2}} A_{5,j}^2 \right)^{1/2}

Remarks
1. SNRM2 is a Cray BLAS (Basic Linear Algebra Subprograms) routine (Cray Research, Inc., 1989b).
Subroutine TBC

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td></td>
<td>Set time-dependent boundary condition values.</td>
</tr>
</tbody>
</table>

**Input**

- **GTBC1, GTBC2**: Time-dependent surface mean flow boundary condition values for the $\xi$ and $\eta$ directions.
- **IT**: Current time step number $n$.
- **ITBEG**: The time level $n$ at the beginning of a run.
- **ITEND**: Final time step number.
- **JBC1, JBC2**: Surface mean flow boundary condition types for the $\xi$ and $\eta$ directions.
- **JTBC1, JTBC2**: Flags for type of time dependency for mean flow boundary conditions in the $\xi$ and $\eta$ directions.
- **NBC**: Dimensioning parameter specifying number of boundary conditions per equation.
- **NEQ**: Number of coupled equations being solved, $N_{eq}$.
- **NOUT**: Unit number for standard output.
- **NTBC**: Number of values in tables for general unsteady boundary conditions.
- **NTBCA**: Time levels at which general unsteady boundary conditions are specified.
- **N1, N2**: Number of grid points $N_1$ and $N_2$, in the $\xi$ and $\eta$ directions.

**Output**

- **FBC1, FBC2**: Point-by-point mean flow boundary condition values for the $\xi$ and $\eta$ directions.
- **GBC1, GBC2**: Surface mean flow boundary condition values for the $\xi$ and $\eta$ directions.

**Description**

Subroutine TBC sets time-dependent mean flow boundary condition values. Two types of time dependency are allowed - general and periodic.

**General Time-Dependent Boundary Conditions**

General time-dependent boundary conditions are set using linear interpolation on an input table of boundary condition values vs. time level. Thus, the boundary condition value is

$$g^{n+1} = g^i + \frac{n + 1 - n^i}{n^i + 1 - n^i} (g^{i+1} - g^i)$$

Here $n$ is the current known time level in the time marching scheme, $g^i$ and $n^i$ represent the input table of boundary condition values vs. time level, and $i$ is the index in the table for which...
\[ n_l^j \leq n + 1 < n_l^{j+1} \]

If \( n + 1 < n_l \), then \( g^{n-1} \) is set equal to the first value in the table, \( g_l \). Similarly, if \( n + 1 > n_l \), where \( N \) is the index of the last entry in the table, then \( g^{n-1} \) is set equal to the last value in the table, \( g_N \).

In Fortran, \( g = \text{GBC1 or GBC2} \), \( g_l = \text{GTBC1 or GTBC2} \), \( n_l = \text{NTBCA} \), and \( N = \text{NTBC} \).

**Time-Periodic Boundary Conditions**

Time-periodic boundary conditions (not to be confused with spatially periodic boundary conditions) are of the form

\[
g^{n+1} = g_l^1 + g_l^2 \sin[g_l^3 (n + 1) + g_l^4]
\]

where \( g_l^1 \) through \( g_l^4 \) are given by the first four elements of \( \text{GTBC1 or GTBC2} \).

**Remarks**

1. An error message is generated and execution is stopped if an invalid type of unsteadiness is requested for the boundary values.
Subroutine TIMSTP

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>ISAMAX</td>
<td>Set computational time step.</td>
</tr>
</tbody>
</table>

**Input**

- **CFL**: CFL number in IDTAU = 1, 2, 5, 6, 8, and 9 options.
- **CFLMIN, CFLMAX**: Minimum and maximum CFL numbers allowed in IDTAU = 2 and 6 options.
- **CHGMAX**: Maximum change in absolute value of the dependent variables over previous time step (or NITAVG – 1 time steps if ICTEST = 2), ΔQmax.
- **CHG1, CHG2**: Minimum and maximum change, in absolute value, that is allowed in any dependent variable before increasing or decreasing Δτ in IDTAU = 2, 4, and 6 options.
- **CP, CV**: Specific heats cₚ and cᵥ at time level n.
- **DT**: Time step Δτ in IDTAU = 3 and 4 options.
- **DTAU**: Old computational time step Δτ.
- **DTF1, DTF2**: Factors multiplying or dividing Δτ if solution changes too slowly or quickly in IDTAU = 2, 4, and 6 options.
- **DTMIN, DTMAX**: Minimum and maximum Δτ allowed in IDTAU = 4 option, or used in IDTAU = 7 option.
- **DXI, DETA**: Computational grid spacing Δξ and Δη.
- **ETAX, ETAY, ETAT**: Metric coefficients ηₓ, ηᵧ (or η, if axisymmetric), and ηᵧ.
- **IDTAU**: Flag for time step selection method.
- **IT**: Current time step number n.
- **ITSEQ**: Current time step sequence number.
- **MU**: Effective coefficient of viscosity μ at time level n.
- **NDTCYC**: Number of time steps per cycle for IDTAU = 7 option.
- **NEQ**: Number of coupled equations being solved, Neq.
- **NOUT**: Unit number for standard output.
- **N1, N2**: Number of grid points N₁ and N₂, in the ξ and η directions.
- **RER**: Reference Reynolds number Reᵣ.
- **RGAS**: Gas constant R.
- **RHO, U, V**: Static density ρ, and velocities u and v, at time level n.
- **T**: Static temperature T at time level n.
- **XIX, XIY, XIT**: Metric coefficients ξₓ, ξᵧ (or ξ, if axisymmetric), and ξᵧ.

**Output**

- **CFL**: New CFL number in IDTAU = 2 and 6 options.
- **DTAU**: New computational time step Δτ.
Description

Subroutine TIMSTP computes the time step size $\Delta t$. The following sections describe the various methods currently available for setting and/or modifying $\Delta t$.

IDTAU = 1

This option sets a global (i.e., constant in space) time step $\Delta t$ equal to the minimum of the values at each grid point computed from the input parameter CFL(ITSEQ). I.e.,

$$
\Delta t = (CFL) \min_{i,j}(\Delta t_{cfl})
$$

where $\Delta t_{cfl}$ is the inviscid CFL limit, given in generalized two-dimensional coordinates as (Shang, 1984).

$$
\Delta t_{cfl} = \left\{ \left| \frac{U}{\Delta \xi} \right| + \left| \frac{V}{\Delta \eta} \right| + \frac{a}{2} \left[ \left( \frac{\xi_x}{\Delta \xi} + \frac{\eta_x}{\Delta \eta} \right)^2 + \left( \frac{\xi_y}{\Delta \xi} + \frac{\eta_y}{\Delta \eta} \right)^2 \right]^{1/2} \right\}^{-1}
$$

Here $U = \xi_t + \xi_u + \xi_v$ and $V = \eta_t + \eta_u + \eta_v$ are the contravariant velocities without metric normalization, and $a = \sqrt{\gamma RT}$ is the speed of sound.

IDTAU = 2

For the first time step, this option is identical to the IDTAU = 1 option. After the first time step, however, CFL is modified to keep $\Delta Q_{max}$, the maximum change in absolute value of the dependent variables, within user-specified limits. The rules used to increase or decrease CFL may be summarized as follows:

- $\Delta Q_{max} < CHG1 \Rightarrow CFL = \min[(DTF1)/(CFL), CFLMAX]$
- $\Delta Q_{max} > CHG2 \Rightarrow CFL = \max[CFL/DTF2, CFLMIN]$
- $\Delta Q_{max} > 0.15 \Rightarrow CFL = CFL/2$

The time step $\Delta t$ is then set using the same formulas as in the IDTAU = 1 option.

IDTAU = 3

This option sets a global (i.e., constant in space) time step $\Delta t$ equal to the input parameter DT(ITSEQ).

IDTAU = 4

For the first time step, this option is identical to the IDTAU = 3 option. After the first time step, however, $\Delta t$ is modified to keep $\Delta Q_{max}$, the maximum change in absolute value of the dependent variables, within user-specified limits. The rules used to increase or decrease $\Delta t$ may be summarized as follows:

- $\Delta Q_{max} < CHG1 \Rightarrow \Delta t = \min[(DTF1)/(CFL), DTMAX]$
- $\Delta Q_{max} > CHG2 \Rightarrow \Delta t = \max[\Delta t/(DTF2), DTMIN]$
- $\Delta Q_{max} > 0.15 \Rightarrow \Delta t = \Delta t/2$

IDTAU = 5

This option sets a local (i.e., varying in space) time step $\Delta t$ computed at each grid point from the input parameter CFL(ITSEQ). I.e., at each grid point,

$$
\Delta t = (CFL) \Delta t_{cfl}
$$

where $\Delta t_{cfl}$ is given above in the description of the IDTAU = 1 option.
**IDTAU = 6**

For the first time step, this option is identical to the IDTAU = 5 option. After the first time step, however, CFL is modified to keep $\Delta Q_{max}$, the maximum change in absolute value of the dependent variables, within user-specified limits. The rules used to increase or decrease CFL are the same as in the IDTAU = 2 option.

**IDTAU = 7**

This option sets a global (i.e., constant in space) time step $\Delta t$ with logarithmic cycling. The formula used is

$$\Delta t = \Delta t_{min} \left( \frac{\Delta t_{max}}{\Delta t_{min}} \right)^{N/(N_{cyc} - 1)}$$

where $N = \text{mod}(n - 1, N_{cyc})$ and $n$ is the current known time level. The time step $\Delta t$ is thus cycled repeatedly between $\Delta t_{min}$ and $\Delta t_{max}$ every $N_{cyc}$ time steps. The values of $\Delta t_{min}$, $\Delta t_{max}$, and $N_{cyc}$ are given by the input parameters DTMIN, DTMAX, and NDTCYC.

**IDTAU = 8**

This option sets a local (i.e., varying in space) time step $\Delta t$ computed at each grid point using the procedure of Knight and Choi (1989). The inviscid CFL limit $\Delta t_{in}$ is first computed separately for each computational coordinate direction. Thus, at each grid point,

$$(\Delta t_{cf})_{\xi} = \left[ \left| \frac{U}{\Delta \xi} \right| + a \frac{\sqrt{\xi_x^2 + \xi_y^2}}{\Delta \xi} \right]^{-1}$$

$$(\Delta t_{cf})_{\eta} = \left[ \left| \frac{V}{\Delta \eta} \right| + a \frac{\sqrt{\eta_x^2 + \eta_y^2}}{\Delta \eta} \right]^{-1}$$

Here $U = \xi_U + \xi_x u + \xi_y v$ and $V = \eta_U + \eta_x u + \eta_y v$ are the contravariant velocities without metric normalization, and $a = \sqrt{\gamma RT}$ is the speed of sound.

A preliminary value of $\Delta t$ is then defined at each grid point using the input parameter CFL(ITSEQ).

$$\Delta t_0 = (\text{CFL}) \min[(\Delta t_{cf})_{\xi}, (\Delta t_{cf})_{\eta}]$$

The final value of $\Delta t$ is then defined at each grid point as

$$\Delta t = \max[\Delta t_0, (\Delta t_{cf})_{\xi}]$$

Knight and Choi found that using this definition for $\Delta t$, rather than simply setting $\Delta t = \Delta t_0$, resulted in faster convergence for problems with refined grid regions. This formulation assumes that flow is generally in the $\xi$ direction.

**IDTAU = 9**

This option is similar to the IDTAU = 8 option. The only difference is a viscous correction added to the definitions of the inviscid CFL limits, similar to that used by Cooper (1987). The inviscid CFL limits are now defined at each grid point as:
\[
(\Delta \tau_{dfs})_\xi = \left[ \frac{U}{\Delta \xi} + a \frac{\sqrt{\xi_x^2 + \xi_y^2}}{\Delta \xi} + \frac{2}{Re} \frac{\mu}{\rho} \frac{\xi_x^2 + \xi_y^2}{(\Delta \xi)^2} \right]^{-1}
\]

\[
(\Delta \tau_{dfs})_\eta = \left[ \frac{V}{\Delta \eta} + a \frac{\sqrt{\eta_x^2 + \eta_y^2}}{\Delta \eta} + \frac{2}{Re} \frac{\mu}{\rho} \frac{\eta_x^2 + \eta_y^2}{(\Delta \eta)^2} \right]^{-1}
\]

The rest of the procedure for computing \(\Delta \tau\) is the same as in the IDTAU = 8 option.

**Remarks**

1. In \(\Delta Q_{max}\), used in the IDTAU = 2, 4, and 6 options, the change in \(E_T\) has been divided by \(R/(y_r - 1) + 1/2\). This is equivalent to dividing the dimensional value \(E_T\) by

\[
E_T = \frac{\rho R T_r}{y_r - 1} + \frac{\rho h_r^2}{2}
\]

This makes the change in total energy the same order of magnitude as the other conservation variables.

2. An error message is generated and execution is stopped if an illegal time step selection option is requested.

3. A warning message is printed with the IDTAU = 2, 4, and 6 options if \(\Delta \tau\) or the CFL number is cut in half because \(\Delta Q_{max} > 0.15\).

4. The Cray search routine ISAMAX is used in computing the maximum value of \(\Delta Q_{max}\) for all the equations.
Subroutine Tremain (CPUREM)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td></td>
<td>Get CPU time remaining for the job.</td>
</tr>
</tbody>
</table>

**Input**

None.

**Output**

| CPUREM | Amount of CPU time remaining, in seconds. |

**Description**

Subroutine Tremain computes the amount of CPU time remaining for the current job, in seconds.

**Remarks**

1. Tremain is a Cray Fortran library routine (Cray Research, Inc., 1989a).
Subroutine TURBBL manages the computation of the effective coefficient of viscosity, second coefficient of viscosity, and coefficient of thermal conductivity using the algebraic eddy viscosity model of Baldwin and Lomax (1978). It is called from MAIN during each step from time level \( n \) to \( n + 1 \), but after the governing flow equations have been solved. The Fortran variables RHO, U, etc., are thus at the \( n + 1 \)
level. The effective viscosity coefficient to be computed will therefore also be at the \( n + 1 \) level. This, of course, becomes the known \( n \) level for the next time step.

The steps involved in computing the effective coefficients are as follows:

1. Initialize the arrays for storing the turbulent viscosity \( \mu_t \) on constant \( \xi \) and \( \eta \) lines to zero.

2. Call \textsc{Vortex} to compute \( |\hat{\Omega}| \), the magnitude of the total vorticity vector.

3. At each \( \xi \) location, compute \( \mu \), due to walls at \( \eta = 0 \) and/or \( \eta = 1 \), or due to a free turbulent flow in the \( \xi \) direction, using steps 3a - 3c. The result will be stored in the Fortran array \textsc{MUT}. If bypassing the calculation on constant \( \xi \) lines, skip to step 4 to compute \( \mu_t \) on constant \( \eta \) lines.
   
   3a. Determine wall locations by checking for zero velocity at the \( \eta \) boundaries, unless wall locations are user-specified via the input \textsc{LWall} or \textsc{iWWall} parameters, or unless boundary types are specified using the \textsc{KBC} parameters.

   3b. Call \textsc{BLOUT1} to compute \( (\mu_t)_{\text{outer}} \), at the current \( \xi \) location, for \( \eta = 0 \) to \( 1 \).

   3c. Call \textsc{BLIN1} to compute \( (\mu_t)_{\text{inner}} \), at the current \( \xi \) location, within the inner region for a solid wall at \( \eta = 0 \) and/or \( \eta = 1 \).

4. At each \( \eta \) location, compute \( \mu_t \), due to walls at \( \xi = 0 \) and/or \( \xi = 1 \), or due to a free turbulent flow in the \( \eta \) direction, using steps 4a - 4c. The result will be stored in the Fortran array \textsc{DUMMY}. If bypassing the calculation on constant \( \eta \) lines, skip to step 5.

   4a. Determine wall locations by checking for zero velocity at the \( \xi \) boundaries, unless wall locations are user-specified via the input \textsc{LWall} or \textsc{iWWall} parameters, or unless boundary types are specified using the \textsc{KBC} parameters.

   4b. Call \textsc{BLOUT2} to compute \( (\mu_t)_{\text{outer}} \), at the current \( \eta \) location, for \( \xi = 0 \) to \( 1 \).

   4c. Call \textsc{BLIN2} to compute \( (\mu_t)_{\text{inner}} \), at the current \( \eta \) location, within the inner region for a solid wall at \( \xi = 0 \) and/or \( \xi = 1 \).

5. If the input is such that the computation of \( \mu_t \) is bypassed in both directions, write an error message and stop.

6. If \( \mu_t \) is being computed on constant \( \xi \) lines only, then \textsc{MUT} = \( \mu_t \), so skip to step 9.

7. If \( \mu_t \) is being computed on constant \( \eta \) lines only, then \textsc{DUMMY} = \( \mu_t \), so set \textsc{MUT} = \textsc{DUMMY} and skip to step 9.

8. If \( \mu_t \) is being computed both on constant \( \xi \) lines and constant \( \eta \) lines, compute a single \( \mu_t \) value at each grid point using the averaging formula presented in equation (9.13) of Volume 1.

9. If specified in the input, modify \( \mu_t \) to account for laminar-turbulent transition using a model based on one given by Cebeci and Bradshaw (1984). This model is described in Section 9.1.4 of Volume 1.

10. Define the necessary effective coefficients as follows:

\[
\begin{align*}
\mu &= \mu_t + \mu_t \\
\lambda &= \lambda_t + \lambda_t \\
\kappa &= k_t + k_t
\end{align*}
\]

where \( \lambda_t = -2\mu_t/3 \), and \( k_t \) is computed using Reynold's analogy as

\[
\kappa_t = \frac{\mu \chi_p}{Pr_t} Pr_r
\]

The turbulent Prandtl number is either a constant specified in the input, or a variable computed using equation (9.19) of Volume 1.
Remarks

1. In the averaging formula used when \( \mu_t \) is computed both on constant \( \zeta \) lines and constant \( \eta \) lines, the Fortran variables \( F_1 \) and \( F_2 \) are

\[
F_1 = \frac{(y_n)_2}{\left[(y_n)_1^2 + (y_n)_2^2\right]^{1/2}}
\]

\[
F_2 = \frac{(y_n)_1}{\left[(y_n)_1^2 + (y_n)_2^2\right]^{1/2}}
\]

If \((y_n)_1\) and \((y_n)_2\) are both close to zero, \( F_1 \) and \( F_2 \) are set equal to \( 1/\sqrt{2} \), which is the limiting value in the above equations as \((y_n)_1\) and \((y_n)_2\) approach zero.

2. The exponent in the definition of \( \gamma_r \) is limited to 20.

3. In the Fortran equation for the effective thermal conductivity, the factor \( PRR = Pr \) is necessary for proper nondimensionalization of \( k_\tau \).

4. The distance used in the formula for \( \gamma_r \) is a straight-line distance from one point to another. It would probably be better to compute a curvilinear distance along the coordinate line.

5. The scratch array DUMMY, from the common block DUMMY1, is used to store the value of the turbulent viscosity along constant \( \eta \) lines. The array is filled in subroutines BLIN2 and BLOUT2.

6. If \( ITXI \) and \( ITETA \) are both zero, indicating the turbulent viscosity computation is to be bypassed for both coordinate directions, an error message is generated and execution is stopped.
Subroutine TURBCH manages the computation of the effective coefficient of viscosity, second coefficient of viscosity, and coefficient of thermal conductivity using the low Reynolds number k-ε two-equation turbulence model of Chien (1982). The k-ε equations are uncoupled from the mean flow equations, lagged in time and solved separately. This allows maximum modularity in turbulence modeling.

For each step from time level $n$ to $n + 1$, the mean flow equations are solved first, using a time step $\Delta \tau$. The k-ε equations are then solved, using $\text{NTKE}$ time steps with a time step size of $\text{TFACT}(\Delta \tau)$.

The steps involved in computing the effective coefficients are as follows:
1. Determine wall locations by checking for zero velocity at the boundaries, unless wall locations are user-specified via the input LWALL or IWALL parameters, or unless boundary types are specified using the KBC parameters.

2. Call YPLUSN to compute the minimum distance to the nearest solid wall and $y^+$. To save storage, the minimum distance is returned in the Fortran variable DUMMY.

3. Call PRODCT to compute the production rate of turbulent kinetic energy. To save storage space, the production rate is returned in the Fortran variable VORT.

4. Call EXECT to advance the $k$-$\varepsilon$ equations in time using a time step of TFACT(Δτ).

5. Repeat steps 3-4 NTKE times.

6. Define the necessary effective coefficients as follows:
   \[ \mu = \mu_i + \mu_t \]
   \[ \lambda = \lambda_i + \lambda_t \]
   \[ k = k_i + k_t \]

   where \( \lambda_i = -2\mu_i/3 \), and \( k_t \) is computed using Reynolds' analogy as
   \[ k_t = \frac{\mu_c}{Pr_t} Pr_t \]

   The turbulent Prandtl number is either a constant specified in the input, or a variable computed using equation (9.19) of Volume 1.

Remarks

1. The scratch array DUMMY, from the common block DUMMY1, is used to store the values of the minimum distance to the nearest wall. The array is filled in subroutine YPLUSN.

2. The Fortran array VORT, from the common block TURB1, is used to store the values of the production rate of turbulent kinetic energy. The array is filled in subroutine PRODCT.

3. For equal mean flow and $k$-$\varepsilon$ time steps, use TFACT = 1/NTKE.
Subroutine UPDATE (S,NVD,NPTSD)

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td></td>
<td>Update flow variables after each ADI sweep.</td>
</tr>
</tbody>
</table>

**Input**

IBASE, ISTEP  
Base index and multiplication factor used in computing one-dimensional index for two-dimensional array.

* IHSTAG  
Flag for constant stagnation enthalpy option.

* ISWIRL  
Flag for swirl in axisymmetric flow.

IV  
Index in the "vectorized" direction, i.

JI  
Inverse Jacobian of the nonorthogonal grid transformation, J⁻¹.

NPTS  
Number of grid points in the sweep direction, N.

NR, NRU, NRV, NRW, NET  
Array indices associated with the dependent variables ρ, ρu, ρv, ρw, and Er.

NVD, NPTSD  
Leading two dimensions for the array S.

RHO, U, V, W, ET  
Static density ρ, velocities u, v, w, and total energy Er at time level n.

S  
Computed solution subvector, ΔQ.

**Output**

RHOL, UL, VL, WL, ETL  
Static density ρ, velocities u, v, and w, and total energy Er at end of current ADI sweep.

**Description**

Subroutine UPDATE computes the primitive flow variables from the dependent variables ΔQ after each ADI sweep. For the first sweep the formulas are

\[
\begin{align*}
\rho^* &= \rho^n + J\Delta\hat{Q}_1^n \\
u^* &= \frac{1}{\rho^n}(\rho^n u^n + J\Delta\hat{Q}_2^n) \\
v^* &= \frac{1}{\rho^n}(\rho^n v^n + J\Delta\hat{Q}_3^n) \\
w^* &= \frac{1}{\rho^n}(\rho^n w^n + J\Delta\hat{Q}_4^n) \\
E_T^* &= E_T^n + J\Delta\hat{Q}_5^n
\end{align*}
\]
where $\Delta \hat{Q}_i$ through $\Delta \hat{Q}_s$ are the dependent variables in delta form for the five governing equations.\textsuperscript{35} For the second ADI sweep, the superscript $\ast$ should be changed to $n + 1$ on $\rho$, $u$, $v$, $w$, and $E_r$, and to $n$ on $\Delta \hat{Q}$.

**Remarks**

1. This subroutine uses one-dimensional addressing of two-dimensional arrays, as described in Section 2.3.

\textsuperscript{35} These formulas are written for the most general case - axisymmetric flow with swirl and non-constant stagnation enthalpy. For simpler cases there may be only three or four equations.
Subroutine UPDTKE

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECT</td>
<td></td>
<td>Update (k) and (\epsilon) after each ADI sweep.</td>
</tr>
</tbody>
</table>

**Input**

- **DXI, DETA**
  - Computational grid spacing \(\Delta \xi\) and \(\Delta \eta\).
- **E**
  - Turbulent dissipation rate \(\epsilon\) at time level \(n\).
- **FBCT1, FBCT2**
  - Point-by-point \(k\)-\(\epsilon\) boundary condition values for the \(\xi\) and \(\eta\) directions.
- **IBCT1, IBCT2**
  - Point-by-point \(k\)-\(\epsilon\) boundary condition types for the \(\xi\) and \(\eta\) directions.
- **JI**
  - Inverse Jacobian of the nonorthogonal grid transformation, \(J^{-1}\).
- **KBCPER**
  - Flags for spatially periodic boundary conditions in the \(\xi\) and \(\eta\) directions.
- **KE**
  - Turbulent kinetic energy \(k\) at time level \(n\).
- **NPT1, NPT2**
  - \(N_1\) and \(N_2\) for non-periodic boundary conditions, \(N_1 + 1\) and \(N_2 + 1\) for spatially periodic boundary conditions in \(\xi\) and \(\eta\).
- **RHO**
  - Static density \(\rho\) at time level \(n\).
- **S**
  - Computed solution subvector \(\Delta \hat{W}\).

**Output**

- **E, EL**
  - Turbulent dissipation rate \(\epsilon\) at time levels \(n + 1\) and \(n\).
- **KE, KEL**
  - Turbulent kinetic energy \(k\) at time levels \(n + 1\) and \(n\).

**Description**

Subroutine UPDTKE computes the primitive flow variables \(k\) and \(\epsilon\) from the dependent variables \(\Delta \hat{W}^n\) after a complete time step. The formulas are

\[
k^{n+1} = \frac{1}{\rho^{n+1}} (\rho^n + 1 k^n + J \Delta \hat{W}_1^n)
\]

\[
\epsilon^{n+1} = \frac{1}{\rho^{n+1}} (\rho^n + 1 \epsilon^n + J \Delta \hat{W}_2^n)
\]

where \(\Delta \hat{W}_1\) and \(\Delta \hat{W}_2\) are the dependent variables in delta form for the \(k\)-\(\epsilon\) equations.

Subroutine UPDTKE also explicitly computes the \(k\) and \(\epsilon\) values on the computational boundaries using the specified boundary conditions, as described below.

*No Change From Initial Conditions, \(\Delta k = 0\) and/or \(\Delta \epsilon = 0\)*

Values for \(k\) and \(\epsilon\) are simply not updated. Therefore, their values on the boundaries remain the same as their initial or restart values.
Specified values, \( k = f \) and or \( \varepsilon = f \)

Values of \( k \) and \( \varepsilon \) are simply set equal to the specified values.

Specified Two-Point Gradient in Coordinate Direction, \( \partial k / \partial \phi = f \) and or \( \partial \varepsilon / \partial \phi = f \)

Applying \( \partial k / \partial \phi = f \) at the \( \xi = 0 \) boundary, and using two-point one-sided differencing, gives

\[
k_{1,j} = k_{2,j} - f \Delta \xi
\]

At the \( \xi = 1 \) boundary,

\[
k_{N_1,j} = k_{N_1 - 1,j} + f \Delta \xi
\]

Analogous equations can easily be written for the \( \eta \) boundaries, and for \( \partial \varepsilon / \partial \phi = f \).

Specified Three-Point Gradient in Coordinate Direction, \( \partial k / \partial \phi = f \) and or \( \partial \varepsilon / \partial \phi = f \)

Applying \( \partial k / \partial \phi = f \) at the \( \xi = 0 \) boundary, and using three-point one-sided differencing, gives

\[
k_{1,j} = \frac{(4k_{2,j} - k_{3,j} - 2f \Delta \xi)}{3}
\]

At the \( \xi = 1 \) boundary,

\[
k_{N_1,j} = \frac{(k_{N_1 - 1,j} - k_{N_1 - 2,j} + 2f \Delta \xi)}{3}
\]

Analogous equations can easily be written for the \( \eta \) boundaries, and for \( \partial \varepsilon / \partial \phi = f \).

**Linear Extrapolation**

Linearly extrapolating from the interior points for \( k \) at the \( \xi = 0 \) boundary gives

\[
k_{1,j} = 2k_{2,j} - k_{3,j}
\]

At the \( \xi = 1 \) boundary,

\[
k_{N_1,j} = 2k_{N_1 - 1,j} - k_{N_1 - 2,j}
\]

Analogous equations can easily be written for the \( \eta \) boundaries, and for linear extrapolation of \( \varepsilon \).

**Remarks**

1. The "no change from initial conditions" boundary condition is applied simply by non-execution of the other boundary conditions.

2. Periodic boundary conditions are updated by setting the values of \( k \) and \( \varepsilon \) at the lower boundary equal to the corresponding values at the upper boundary.
Subroutine VORTEX

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td></td>
<td>Compute magnitude of total vorticity.</td>
</tr>
<tr>
<td>TURBBL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YPLUSN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input

- DXI, DETA: Computational grid spacing $\Delta \xi$ and $\Delta \eta$.
- ETAX, ETAY: Metric coefficients $\eta_x$ and $\eta_y$ (or $\eta_r$ if axisymmetric.)
- N1, N2: Number of grid points $N_1$ and $N_2$ in the $\xi$ and $\eta$ directions.
- U, V, W: Velocities $u$, $v$, and $w$.
- XI, XJ: Metric coefficients $\xi_x$ and $\xi_y$ (or $\xi_r$ if axisymmetric.)
- Y: Radial coordinate $r$ for axisymmetric flow.

Output

- VORT: Total vorticity magnitude.

Description

Subroutine VORTEX computes the magnitude of the total vorticity vector. For two-dimensional planar flow this is defined as

$$|\vec{\Omega}| = \left| \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right|$$

and for axisymmetric flow,

$$|\vec{\Omega}| = \left[ \left( \frac{\partial w}{\partial r} + \frac{w}{r} \right)^2 + \left( \frac{\partial w}{\partial x} \right)^2 \right]^{1/2}$$

Note that, for flow without swirl, the definition for axisymmetric flow is the same as for two-dimensional planar flow.

Using the chain rule, these can be rewritten in generalized nonorthogonal coordinates. For two-dimensional planar flow,

$$|\vec{\Omega}| = |(\xi_x v_x + \eta_x v_\eta) - (\xi_y u_x + \eta_y u_\eta)|$$

and for axisymmetric flow,

$$|\vec{\Omega}| = \left[ \left( \xi_x w_x + \eta_x w_\eta + \frac{w}{r} \right)^2 + \left( \xi_x w_x + \eta_x w_\eta \right)^2 + \left( \xi_x v_x + \eta_x v_\eta - \xi_x u_x - \eta_x u_\eta \right)^2 \right]^{1/2}$$

At interior points, the centered difference formula presented in Section 5.0 of Volume 1 is used to numerically compute the derivatives in the above equations. At boundary points, second-order one-sided difference formulas are used.
### Subroutine YPLUSN

<table>
<thead>
<tr>
<th>Called by</th>
<th>Calls</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITC</td>
<td>VORTEX</td>
<td>Compute the distance to the nearest solid wall.</td>
</tr>
<tr>
<td>KEINIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TURBCH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Input

- **LWALL1, LWALL2** Flags specifying wall locations for \( \xi \) and \( \eta \) boundaries.
- **MU** Effective coefficient of viscosity \( \mu \).
- **N1, N2** Number of grid points \( N_1 \) and \( N_2 \), in the \( \xi \) and \( \eta \) directions.
- **RER** Reference Reynolds number \( Re \).
- **RHO** Static density \( \rho \) at time level \( n \).
- **VORT** Total vorticity magnitude.
- **X, Y** Cartesian coordinates \( x \) and \( y \), or cylindrical coordinates \( x \) and \( r \).

#### Output

- **DUMMY** Distance to the nearest solid wall.
- **YPLUSD** Nondimensional distance \( y^* \) from the nearest solid wall.

#### Description

Subroutine YPLUSN computes the minimum distance to the nearest solid wall and \( y^* \) for every grid point in the computational domain. The steps involved are as followed:

1. Call VORTEX to compute total vorticity magnitude.
2. For every grid point in the computational domain,
   3. Compute the shortest distance to each solid wall, and the corresponding wall values of the total vorticity magnitude, laminar viscosity, and density.
   4. Identify the nearest solid wall and select the corresponding minimum distance to the wall \( y_n \), the wall total vorticity magnitude \( |\Omega_{wall}| \), the wall laminar viscosity \( \mu_{wall} \), and the wall density \( \rho_{wall} \).
5. Compute \( y^* \) using

\[
y^* = y_n \sqrt{\frac{Re \  |\Omega_{wall}| \rho_{wall}}{\mu_{wall}}}\]

#### Remarks

1. The scratch array DUMMY, from the common block DUMMY1, is used to store the minimum distance to the nearest solid wall.
REFERENCES


**4. TITLE AND SUBTITLE**

Proteus Two-Dimensional Navier-Stokes Computer Code-Version 2.0
Volume 3-Programmer’s Reference

**6. AUTHOR(S)**

Charles E. Towne, John R. Schwab, and Trong T. Bui

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

**8. PERFORMING ORGANIZATION REPORT NUMBER**

E–8108

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
Washington, D.C. 20546–0001

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

NASA TM–106339

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

Unclassified - Unlimited
Subject Category 34

**13. ABSTRACT (Maximum 200 words)**

A computer code called *Proteus 2D* has been developed to solve the two-dimensional planar or axisymmetric, Reynolds-averaged, unsteady compressible Navier-Stokes equations in strong conservation law form. The objective in this effort has been to develop a code for aerospace propulsion applications that is easy to use and easy to modify. Code readability, modularity, and documentation have been emphasized. The governing equations are solved in generalized nonorthogonal body-fitted coordinates, by marching in time using a fully-coupled ADI solution procedure. The boundary conditions are treated implicitly. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. Turbulence is modeled using either an algebraic or two-equation eddy viscosity model. The thin-layer or Euler equations may also be solved. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used. Several time step options are available for convergence acceleration. The documentation is divided into three volumes. This is the Programmer’s Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

**14. SUBJECT TERMS**

Navier-Stokes; Computational fluid dynamics, Viscous flow; Compressible flow