WIND User's Guide*

The NPARC Alliance

NASA Glenn Research Center
Cleveland, Ohio

USAF Arnold Engineering Development Center
Tullahoma, Tennessee

*This document describes the use of WIND Version 5.0, and was created December 5, 2007. It was originally based on the 24 Jan 1997 version of the *NASTD User's Guide*, by R. H. Bush, M. Mani, T. R. Michal, and W. W. Romer of McDonnell Douglas Aerospace. Please send documentation suggestions/corrections to Charlie Towne at *towne@grc.nasa.gov*. Questions about the WIND code itself or the NPARC Alliance should be sent to *nparc-support@info.arnold.af.mil*, (931) 454-7455

Contents

1 Introduction **1**
- 1.1 Mathematical Model . . . 1
- 1.2 Geometry and Mesh Description . . . 1
- 1.3 Numerical Technique . . . 1
- 1.4 Coding . . . 2

2 Tutorial **3**
- 2.1 Gather Information . . . 3
  - 2.1.1 Geometry . . . 3
  - 2.1.2 Flow Conditions . . . 4
  - 2.1.3 Boundary Conditions . . . 4
- 2.2 Create the Computational Grid . . . 5
  - 2.2.1 General Requirements . . . 5
  - 2.2.2 Creating the Common Grid (*.cgd*) File . . . 6
  - 2.2.3 Test Case 4 Grid . . . 6
- 2.3 Define the Input . . . 11
  - 2.3.1 Descriptive Header and Comments . . . 11
  - 2.3.2 Boundary Conditions . . . 12
  - 2.3.3 Flow and Initial Conditions . . . 12
  - 2.3.4 Physical Model Controls . . . 12
  - 2.3.5 Numerical Model Controls . . . 13
  - 2.3.6 Test Case 4 Input . . . 15
- 2.4 Run the Code . . . 19
  - 2.4.1 The *wind* Script . . . 19
  - 2.4.2 Parallel Operation . . . 19
  - 2.4.3 Running Test Case 4 . . . 19
- 2.5 Monitor Convergence . . . 22
  - 2.5.1 Test Case 4 Convergence . . . 22
- 2.6 Examine the Results . . . 25
  - 2.6.1 Using CFPOST . . . 26
  - 2.6.2 Test Case 4 Results . . . 26
- 2.7 Summary . . . 27

3 Geometry and Flow Physics Modeling **29**
- 3.1 Symmetry Considerations . . . 29
  - 3.1.1 Three-Dimensional Cases . . . 29
  - 3.1.2 Two-Dimensional Cases . . . 29
  - 3.1.3 Area Variation (Quasi-3D) Cases . . . 29
  - 3.1.4 Axisymmetric Cases . . . 29
- 3.2 Euler and Navier-Stokes Equations . . . 30
  - 3.2.1 Freestream Conditions . . . 30
  - 3.2.2 Reynolds Number Considerations . . . 30
  - 3.2.3 Mass Flow in Two-Dimensional Calculations . . . 32
  - 3.2.4 Mass Flow and Grid Areas . . . 32
  - 3.2.5 Heat Transfer . . . 33
  - 3.2.6 Viscosity . . . 33
- 3.3 Turbulence Models . . . 33
  - 3.3.1 Algebraic Models . . . 33
  - 3.3.2 One-Equation Models . . . 34

3.3.3 Two-Equation Models . . . 34
3.3.4 Transition Specification . . . 35
3.4 Gas Models . . . 35
3.5 Other Models . . . 35
3.5.1 Actuator Disks . . . 35
3.5.2 Screens . . . 35
3.5.3 Vortex Generators . . . 35
3.6 Flowfield Initialization . . . 36
3.6.1 User-Specified Initialization . . . 36
3.6.2 Boundary Layer Initialization . . . 37
3.6.3 Reinitialization . . . 37

4 Numerical Modeling 39
4.1 Iterations and Cycles . . . 39
4.1.1 "Turning Off" Zones . . . 39
4.2 Grid Considerations . . . 39
4.2.1 Grid Sequencing . . . 39
4.2.2 Thin-Shear-Layer Calculations . . . 40
4.3 Explicit Operator . . . 40
4.3.1 Explicit Smoothing . . . 40
4.3.2 Explicit Boundary Damping . . . 41
4.3.3 Total-Variation-Diminishing (TVD) Operator . . . 41
4.4 Implicit Operator . . . 41
4.4.1 Implicit Boundaries . . . 41
4.5 Time Step . . . 42
4.5.1 CFL Number . . . 42
4.5.2 Runge-Kutta Time Step . . . 43
4.5.3 Global Newton Iteration . . . 43
4.5.4 Time-Accurate Solutions . . . 43
4.6 Convergence Acceleration . . . 44

5 Boundary Conditions 45
5.1 Explicit and Implicit Boundary Conditions . . . 45
5.2 Boundary Condition Types . . . 45
5.3 Wall Boundary Conditions . . . 46
5.3.1 Inviscid Wall . . . 46
5.3.2 Viscous Wall . . . 46
5.3.3 Bleed . . . 47
5.4 Flow Interface Boundary Conditions . . . 47
5.4.1 Freestream . . . 47
5.4.2 Arbitrary Inflow . . . 48
5.4.3 Outflow . . . 48
5.5 Grid Topology Boundary Conditions . . . 49
5.5.1 Reflection . . . 49
5.5.2 Self-Closing . . . 49
5.5.3 Singular Axis . . . 50
5.5.4 Pinwheel . . . 50
5.6 Zonal Interface Boundary Conditions . . . 50
5.6.1 Coupled . . . 50
5.6.2 Chimera . . . 51
5.7 Miscellaneous Boundary Conditions . . . 51

5.7.1 Undefined . . . 52
5.7.2 Frozen . . . 52

6 Convergence Monitoring **53**
6.1 Residuals . . . 53
6.2 Integrated Flowfield Quantities . . . 53
6.3 History Tracking of Flow Data . . . 54

7 Files **55**
7.1 Input Data File (*.dat*) . . . 55
7.2 Grid File (*.cgd*) . . . 55
7.3 Flow File (*.cfl*) . . . 56
7.4 Global Newton File (*.cfk*) . . . 56
7.5 Boundary Data File (*.tda*) . . . 56
7.6 Time History File (*.cth*) . . . 57
7.7 List Output File (*.lis*) . . . 57
7.8 WIND Stop File (*NDSTOP*) . . . 57
7.9 Temperature and Transition Specification Files . . . 58
7.10 Chemistry Files (*.chm*) . . . 58
7.10.1 Header . . . 58
7.10.2 Thermodynamic Coefficients . . . 59
7.10.3 Finite Rate Coefficients . . . 60
7.10.4 Transport Coefficients . . . 63
7.11 Reserved Files . . . 64

8 Scripts **65**
8.1 *wind* — Run WIND Code . . . 65
8.2 *wind_post* — Perform Post-Processing . . . 68
8.3 *windver* — Get WIND Version Number . . . 70
8.4 *windrun* — Quick WIND run . . . 70
8.5 *windmp* — Run on Multi-Processor . . . 71
8.6 *runtest, runtestsuite* — Run WIND Test Case(s) . . . 71

9 Parallel Processing **75**
9.1 Multi-Processing Control File . . . 75
9.2 Multi-Processing Script Considerations . . . 78
9.3 Multiple Parallel Jobs . . . 79
9.4 Hints . . . 79

10 Keyword Reference **81**
10.1 Text Conventions . . . 81
10.2 Zone Selection . . . 81
10.3 Keyword Details . . . 82
ACCELERATE — Convergence acceleration (block) . . . 83
ACTUATOR | SCREEN — Discontinuous change across a zone boundary (block) . . . 85
ARBITRARY INFLOW — Arbitrary inflow (block) . . . 90
AXISYMMETRIC | AXI-SYM — Axisymmetric flow . . . 99
BLEED — Bleed region flow rate . . . 100
BLOW — Inject vectored flow over a selected region . . . 102
BL_INIT — Boundary layer initialization . . . 105
BOUNDARY-DAMP | BDAMP — Boundary damping (block) . . . 106
BOUNDARY TVD — Boundary total variation diminishing operator flag . . . 107

CFL# — CFL/time step specification . . . 108
CGNSBASE — Use CGNS files . . . 111
CHEMISTRY — Chemistry model selection (block) . . . 112
/ — Comment lines . . . 115
COMPRESSOR FACE — Outflow boundaries, compressor face . . . 116
CONVERGE — Controls convergence . . . 119
COUPLING — Zone coupling mode specification . . . 120
CROSSFLOW — Crossflow CFL factor . . . 121
CYCLES — Number of solution cycles . . . 122
DOWNSTREAM MACH — Outflow boundaries, Mach number . . . 123
DOWNSTREAM PRESSURE — Outflow boundaries, pressure . . . 124
DQ — $\Delta\mathbf{Q}$ limiter . . . 128
END — Termination . . . 129
FIXER — Instability smoothing . . . 130
FREESTREAM — Freestream conditions . . . 131
FRINGE — Solution mode at fringe points . . . 132
GAS — Gas property specification . . . 133
GRAVITY — Add gravity body forces . . . 134
GRID LIMITER — Grid limiting capability . . . 135
HISTORY — Time history flowfield variable tracking . . . 136
HLLE — HLLE scheme anti-diffusion terms . . . 141
HOLD — Hold conditions at freestream inflow boundaries . . . 142
IMPLICIT — Implicit operator control . . . 143
IMPLICIT BOUNDARY — Implicit boundary conditions . . . 145
INCLUDE — Include a file in the standard input . . . 146
ITERATIONS — Set number of iterations per cycle . . . 147
LOADS — Flowfield integration (block) . . . 148
MARCHING — Parabolized Navier-Stokes algorithm . . . 151
MASS FLOW — Outflow boundaries, mass flow . . . 152
MFD — Magneto-Fluid Dynamics Model (block) . . . 155
MOVING WALL — Specify moving wall boundaries . . . 160
NAVIER-STOKES ITERATIONS — Navier-Stokes sub-iterations . . . 161
NEWTON — Use Global Newton time stepping . . . 162
OUTFLOW NON-REFLECTING — Outflow boundaries, non-reflecting . . . 163
PERIODIC — Periodic boundaries . . . 164
REINITIALIZE — Reinitialize selected flowfield zones on restart . . . 165
RELAX COUPLING — Set zone coupling relaxation factor . . . 166
REL-ROT-ZONE — Relative rotating zones (block) . . . 167
RESTART | START — Begin run in specified zone . . . 172
RHS — Explicit operator control . . . 173
ROLL — Specify roll about one of the coordinate axes . . . 175
ROTATE — Perform calculation in a rotating frame of reference . . . 176
SEQUENCE — Grid sequencing control . . . 177
SMOOTHING — Add dissipation to explicit operator . . . 178
SPAWN — Run an external process from WIND . . . 179
STAGES — Multi-stage time stepping . . . 181
TDA_INVALID — Rewrite boundary data to *.tda* file . . . 182
TEST — Non-production test options . . . 183
TSL | THIN SHEAR LAYER — Thin shear layer option . . . 184
TTSPEC — Wall temperature and transition (block) . . . 185
TURBULENCE — Turbulence model selection . . . 188

TVD — Total Variation Diminishing operator flag 198
VORTEX GENERATOR — Vortex generator model (block) 199
VISCOSITY — Specification of viscosity law 203
WALL FUNCTION — Specify the use of wall functions 204
WALL SLIP — Iterations until no slip 205
WALL TEMPERATURE — Specify wall temperature 206

11 Test Options **207**

References **221**

1 Introduction

This manual describes the operation and use of the WIND code, a computational platform which may be used to numerically solve various sets of equations governing physical phenomena. WIND represents a merger of the capabilities of three older CFD codes — NASTD (the primary flow solver at McDonnell Douglas, now part of Boeing), NPARC (the original NPARC Alliance flow solver), and NXAIR (an AEDC code used primarily for store separation problems).[1] Currently, the code supports the solution of the Euler and Navier-Stokes equations of fluid mechanics, along with supporting equation sets governing turbulent and chemically reacting flows.

WIND uses multi-zone computational grids, and is capable of computing solutions on a wide variety of grids. Because WIND is written to accommodate arbitrary grid topologies and boundary condition combinations, it may be used to obtain solutions about most of the geometric configurations for which a grid can be generated.

The multi-zone approach to flow solutions makes it possible to decompose virtually any configuration into a number of manageable subregions, or zones. Zonal connectivity information is computed using a pre-processing Grid MANipulation (GMAN) code, and stored in the grid file used by WIND. During the course of a solution, WIND maintains continuity in flow properties across zone boundaries through a process known as zone coupling (Romer and Bush, 1993).

1.1 Mathematical Model

All terms are retained in the governing equations, including secondary flow, reversed flow convection, pressure gradients normal to a wall, streamwise diffusion, and unsteady flow. All heat transfer terms are retained. Several algebraic, one-equation, and two-equation turbulence models are available. Transition may be specified through the use of an external file. Modification of the effective heat transport coefficient due to turbulence is linked to the momentum diffusion coefficient by a turbulent Prandtl number, which is taken to be constant.

The fluid may be treated as a thermally and calorically perfect gas, a thermally perfect gas, equilibrium air, or a mixture undergoing a finite rate chemical reaction. For ideal gas, conventional values are given to the gas constant (R) and the ratio of specific heats (γ), or they may be specified. Effects of gravity on the fluid (i.e., stratification) are not included.

1.2 Geometry and Mesh Description

WIND uses externally generated computational grids. Therefore, all geometric input and capability depend on the grid generator. WIND has no geometric input. All analyses must be preceded by a grid generation run.

1.3 Numerical Technique

The solution is executed iteratively on the computational mesh. The flow equations are evaluated using second-order-accurate finite differences. The partial differential equations are modeled in their conservative form. Explicit terms are computed using either upwind or central differencing, and

[1] WIND is a product of the NPARC Alliance, a partnership between the NASA Glenn Research Center (GRC) and the USAF Arnold Engineering Development Center (AEDC) dedicated to the establishment of a national, applications-oriented flow simulation capability. The Boeing Company has also been closely associated with the Alliance since its inception, and represents the interests of the NPARC User's Association.

their order may be controlled through the use of keywords in the input data file. The implicit terms are computed using either an approximately factored or four-stage Runge-Kutta scheme, or they may be disabled altogether. A Global Newton iteration scheme is also available, and may be used for unsteady flows with large time scales or as a convergence acceleration technique for steady flows.

1.4 Coding

WIND is coded in the Fortran 77, Fortran 90, and C programming languages. The production version of the code is supported on a variety of systems from Silicon Graphics, Hewlett-Packard, Sun, and Cray.

2 Tutorial

This section is intended primarily for new users to demonstrate the simulation process using WIND. More experienced users may find this section useful as a road map through the simulation process and to help demonstrate new features. The approaches presented here are by no means unique, and detailed information is excluded by design. The user is referred to later sections of this User's Guide for more detailed information on the various aspects of running the WIND code, and in particular to Section 10 for more in-depth discussions on the choices available for each input keyword.

The approach taken here is to discuss the flow simulation process using as an example a simple subsonic internal flow in a diverging duct. The various files discussed in this section may be downloaded from the WIND documentation WWW site, at http://www.grc.nasa.gov/WWW/winddocs/user/tutorial.html#tutorial:downloading.

While it is clearly impossible to demonstrate every option in WIND with a single application, the basic mechanics of using the WIND code are demonstrated with this case. Additional abbreviated examples are also provided in Section 10 for specific keywords. For details of the flow simulation process for more complex cases, please see the various example applications accessible from the WIND Validation home page at `http://www.grc.nasa.gov/WWW/wind/valid/validation.html`.

The solution process using any conventional time-marching Navier-Stokes code is basically the same, and may be divided into the following steps:

- Gather information
- Create the computational grid
- Define the input
- Run the code
- Monitor convergence
- Examine the results

The mechanics of doing each of these steps may vary from code to code, however. The following sections describe how these steps are typically accomplished when using the WIND code.

2.1 Gather Information

As for any project, the first step is to gather all of the information required to completely specify the problem to be analyzed. Of course, as the flow simulation process proceeds, missing information will become apparent. The required information can be divided into three major categories — geometry, flow conditions, and boundary conditions. It is also important to understand the ultimate goal of the simulation. For example, is an accurate drag prediction required? Or, is lift required, but only to within 5%? The answers to these types of questions will determine the detail of the input information required to provide the necessary level of detail and accuracy in the solution.

2.1.1 Geometry

The more geometric details that can be determined for the target application, the more likely the results will provide an accurate simulation of the flow field. This is not to say that all geometric

components must be modeled. Resolving fine geometric details of a configuration requires more grid points, and, as a result, longer run times. The level of detail to which the geometry must be modeled depends on the type of results required and the acceptable turn-around time.

The geometry of the Test Case 4 example is shown in Figure 1. The duct is 8 inches long, and the entrance and exit heights are 1 inch and 2 inches, respectively.

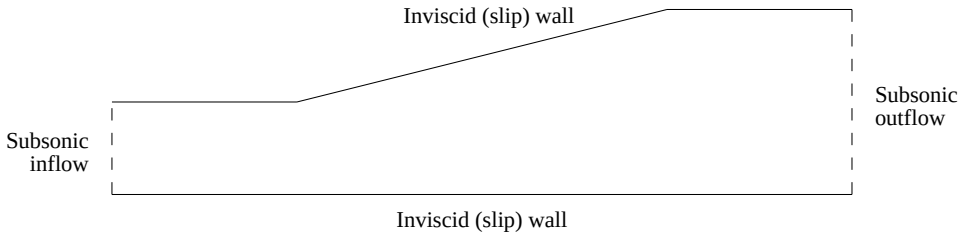


Figure 1: Test case geometry

The desired result from this calculation is the pressure distribution along the upper and lower walls, and the mass flow rate to within 10%. Thus, resolving the sharp surface gradients at the corners is not necessary. Other detailed geometric features such as weld joints, for example, are also not modeled. If the fine details of the boundary layer in the vicinity of the joints were important, then a significantly more detailed geometry description would be required.

2.1.2 Flow Conditions

In addition to the geometric information, flow conditions are also required, and are used to set the reference conditions used in non-dimensionalizing the governing equations solved by the WIND code. As with the geometric information, the simulation results will only be as good as the flow condition information provided by the user. Flow conditions should be specified that are representative of the flow being solved, so that the nondimensional variables used in the code are on the order of 1.0. A good choice is the inlet conditions for internal flows, and the freestream conditions for external flows.

For this example case, the inlet Mach number, total pressure, and total temperature are 0.78, 15 psi, and 600 °R, respectively. Starting from these conditions, the Reynolds number based on the duct height may be computed as 3.023×10^5.

2.1.3 Boundary Conditions

Information is also required at boundaries that are at the "outer edges" of the computational domain. These boundary conditions are used to model the interaction between the flow inside the computational domain and surfaces or flows outside the domain. In fact, the boundary conditions are perhaps the most important factor influencing the accuracy of the flow computation.

Conditions at flow interface boundaries — boundaries between flows inside and outside the computational domain such as inflow, outflow, and freestream boundaries — must be known to the level of accuracy required by the simulation. For example, if flow rates are required to within 0.1%, even slight variations in total pressure at the inflow boundary must be specified. The number of conditions to be specified at a flow interface boundary depends on whether the flow is entering or leaving the computational domain, and whether it is subsonic or supersonic.

Information must also be specified at surface interface boundaries, such as solid walls and bleed regions. Simply specifying the type of boundary, such as an adiabatic no-slip wall, is often sufficient. Additional information may also be required, though, such as the wall temperature. The level of detail that is needed for this information is determined, as discussed above, by the level of detail and accuracy required in the results.

Note that other types of boundaries may be present within the overall computational domain, that are not at the "outer edges." Multi-zone problems will have zonal interface boundaries. Some configurations will also have boundaries resulting from the grid topology, such as self-closing and singular axis boundaries. These types of boundaries need only be labeled.

For Test Case 4 the flow at both the inflow and outflow planes will be subsonic. Three conditions are needed at the inflow boundary, and one is needed at the outflow boundary. At the inflow boundary, uniform flow is specified, with total pressure and temperature equal to the inlet values of 15 psi and 600 °R. Since extreme accuracy in the solution is not needed, constant total conditions at the inflow are sufficient. At the outflow boundary, the exit static pressure is set to 14.13 psi. The Reynolds number for Test Case 4 is large enough that the boundary layers will have little influence on the pressure distribution within the duct. The upper and lower boundaries are therefore specified as inviscid (slip) walls.

In this tutorial, the procedure used to set boundary conditions when running WIND is discussed in Section 2.3.2. Additional details on all the boundary conditions available in the WIND code are presented in Section 5.

2.2 Create the Computational Grid

WIND uses externally-generated structured grids. The grids for all the zones must therefore be created before running the WIND code. The geometry of the application governs the overall shape of the boundaries, but the approach to gridding the flow field is not unique.

2.2.1 General Requirements

The WIND flow simulator provides considerable flexibility. Grid lines can conform to complex shapes or may pass through regions not in the flow field. The grid may be divided into zones to conform to the geometry better, to allow grid embedding (i.e., zones with finer grids in regions of high gradients like boundary layers), and/or to allow parallel computation. These zones may be abutting or overlapping, and overlapping grids may be single- or double-fringed.

WIND requires structured grids, in which the indices (i, j, k) represent a curvilinear coordinate system, and physical Cartesian coordinates (x, y, z) are defined for each integer combination of indices. The handedness of both the physical and curvilinear coordinate systems is required to be the same at all points in the grid, i.e., both must be either left-handed or right-handed. Additionally, at least three grid points must fall between any two grid lines which represent a boundary within the computational domain. For example, if the $k = 1$ boundary represents a solid surface and an adjacent k boundary represents a symmetry plane, the symmetry plane must be at $k = 5$ or higher, so that the three points $k = 2$, 3, and 4 (at least) lie between the two boundaries.

The method used to create the grid is completely up to the user. For complex geometries, a sophisticated grid generation program is normally used. For very simple geometries, it may be easier to write a short program that constructs a grid using algebraic techniques.

2.2.2 Creating the Common Grid (*.cgd*) File

The computational grid used by WIND for a particular case is stored in a *Common Grid* (*.cgd*) file, so named because the file is formatted according to Boeing's Common File Format.[2] In this file are stored the (x, y, z) coordinates of all the grid points, zone coupling information, and grid units and scaling data.[3]

Since some grid generation codes do not produce *.cgd* files directly, a separate utility called *cfcnvt* is included with WIND that may be used to convert a variety of file formats, including PLOT3D files, to Common Files. A typical procedure is thus to first store the grid file as a PLOT3D xyz file, which is an available option in most general-purpose grid generation codes, and then convert it to a *.cgd* file using *cfcnvt*.

Zone coupling information is then added to the *.cgd* file using a pre-processing utility called GMAN. This is typically done at the same time as the boundary condition types are defined, and is discussed in Section 2.3.2. GMAN may then be used to examine the *.cgd* file, assessing grid quality and listing information about the points and zones in the grid. GMAN may also be used to generate the flowfield (i.e., interior) grid itself, given the grids on the zonal boundaries. The *GMAN User's Guide* contains descriptions of these capabilities, and others.

2.2.3 Test Case 4 Grid

The computational grid for Test Case 4 was constructed by simple algebraic techniques using the following program, called *case4.mesh.f*, which creates a PLOT3D xyz file in 2-D unformatted multi-zone form to Fortran unit 2, without an `iblank` array.

```
      Program mesh
c
c-----Purpose:  This subroutine computes a 2-D three-zone grid for a
c               simple diffuser, one of the test cases supplied with the
c               WIND computer code.  The grid is written to a file in
c               PLOT3D 2-D unformatted multi-zone form, without
c               iblank'ing.
c
c-----Called by:
c
c-----Calls:
c
      Implicit none
c-----Parameter statements
      Integer IDIM,JDIM    ! Max dimensions
      Integer NBLKS        ! Number of blocks
      Parameter (IDIM = 33, JDIM = 11)
      Parameter (NBLKS = 3)
c-----Local variables:
      Integer i,j          ! Indices in x and y directions
      Integer iblk         ! Current block number
      Integer imax(NBLKS),jmax(NBLKS)   ! Block grid sizes
      Real dx(NBLKS),dy(NBLKS)        ! Non-dim grid increments in blks
```

[2] WIND also supports the use of CGNS files for the grid and flow solution, using the `CGNSBASE` keyword. This tutorial, however, uses common files.

[3] See the *Common File User's Guide* for details about the internal structure of common files.

```
      Real x(IDIM,JDIM),y(IDIM,JDIM),z(IDIM,JDIM)   ! Grid coordinates
      Real xdiff1,xdiff2   ! x at start/end of diffuser section
      Real xloc            ! Local x coordinate
      Real xmax            ! x at end of duct
      Real xstrt(NBLKS),xend(NBLKS)   ! Non-dim x limits of blocks
      Real ymax            ! Max y at x = xloc
      Real ymax1,ymax2     ! Max y at start/end of diffuser section
      Real yslope          ! Slope of diffuser upper wall
      Real ystrt(NBLKS),yend(NBLKS)   ! Non-dim y limits of blocks
c
c-----Define geometric parameters
c
      Data xdiff1,xdiff2,xmax /2.0, 6.0, 8.0/
      Data ymax1,ymax2 /1.0, 2.0/
c
c-----Set relative sizes and grid increments for each block
c
c-----Block 1
      imax(1) = 17
      jmax(1) =  6
      xstrt(1) = 0.0
      ystrt(1) = 0.0
      xend (1) = 0.5
      yend (1) = 0.5
      dx(1) = (xend(1) - xstrt(1))/(imax(1) - 1)
      dy(1) = (yend(1) - ystrt(1))/(jmax(1) - 1)
c-----Block 2
      imax(2) = 33
      jmax(2) = 11
      xstrt(2) = 0.0
      ystrt(2) = yend(1)
      xend (2) = xend(1)
      yend (2) = 1.0
      dx(2) = (xend(2) - xstrt(2))/(imax(2) - 1)
      dy(2) = (yend(2) - ystrt(2))/(jmax(2) - 1)
c-----Block 3
      imax(3) = 17
      jmax(3) = 11
      xstrt(3) = xend(1)
      ystrt(3) = 0.0
      xend (3) = 1.0
      yend (3) = 1.0
      dx(3) = (xend(3) - xstrt(3))/(imax(3) - 1)
      dy(3) = (yend(3) - ystrt(3))/(jmax(3) - 1)
c
c-----Open grid file, write header info
c
      Open (unit=2, file='fort.2', form='unformatted')
      Write (2) NBLKS
      Write (2) (imax(iblk),jmax(iblk),iblk=1,NBLKS)
c
```

```
c-----Construct the grid
c
      yslope = (ymax2 - ymax1)/(xdiff2 - xdiff1)   ! Slope of diff wall
      Do 100 iblk = 1,NBLKS       ! Loop over no. of blocks
         Do 50 i = 1,imax(iblk)   ! Loop over no. of x points
c-----------Compute x coordinates
            xloc = (xstrt(iblk) + dx(iblk)*(i-1))*xmax
            Do 10 j = 1,jmax(iblk)   ! Loop over no. of y points
               x(i,j) = xloc
10          Continue
c-----------Get local max y, then compute y coordinates
            If (xloc .le. xdiff1) then        ! Upstream of diffuser
               ymax = ymax1
            Else if (xloc .ge. xdiff2) then   ! Downstream of diffuser
               ymax = ymax2
            Else                              ! In diffuser
               ymax = ymax1 + yslope*(xloc-xdiff1)
            End if
            Do 20 j = 1,jmax(iblk)   ! Loop over no. of y points
               y(i,j) = (ystrt(iblk) + dy(iblk)*(j-1))*ymax
20          Continue
50       Continue
c
c-----Write the grid file in PLOT3D xyz format
c
         Write (2) ((x(i,j),i=1,imax(iblk)),j=1,jmax(iblk)),
     &             ((y(i,j),i=1,imax(iblk)),j=1,jmax(iblk))
100   Continue
      Stop
      End
```

Figure 2 shows the resulting grid. Three zones are used, with grid sizes of 17×6, 33×11, and 17×11, respectively.[4]

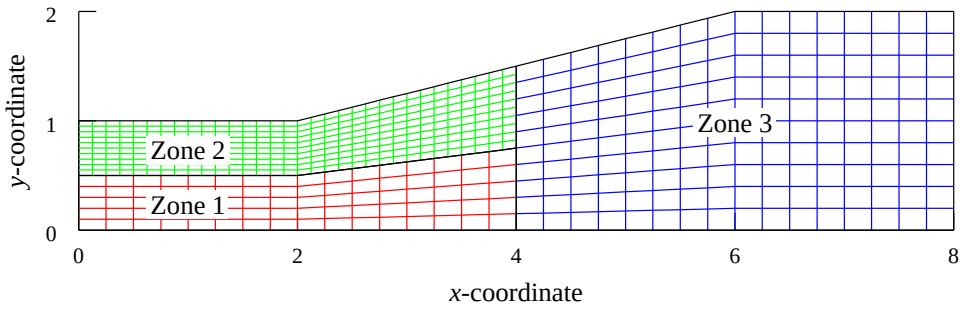


Figure 2: Test case grid

The grid file written by the above program, named *case4.xyz*, was converted to a *.cgd* file named *case4.cgd* using *cfcnvt*, as shown in the following runstream. Lines in slanted type were typed by

[4]This geometry is simple enough that a single-zone grid would be sufficient, but a three-zone grid is used for illustrative purposes.

the user.

```
cfcnvt
 ****************************************************************************
 * Warning: This software contains technical data whose export is           *
 * restricted by the Arms Export Control Act (Title 22, U.S.C., Sec 2751,   *
 * et seq.) or Executive Order 12470. Violation of these export-control     *
 * laws is subject to severe criminal penalties. Dissemination of this      *
 * software is controlled under DoD Directive 5230.25 and AFI 61-204.       *
 ****************************************************************************
          ******* Common File Convert Utilities ******
 CFCNVT - Version 1.45 (last changed 2001/12/14 18:24:56)

   0: Exit program
   2: Import   a Common File
   3: Compress a Common File
   4: Break Common File into multiple transfer files
   5: Combine multiple transfer files into Common File
   6: Append one Common File to another
   7: Convert Common File binary to a text file
   8: Convert Common File text to a binary file
  11: Convert PLOT3D/Pegsus file to Common File
  12: Convert GASP     file to Common File
  13: Convert OVERFLOW file to Common File
  14: Convert Common File to OVERFLOW file
  15: Convert CFPOST GPU file to Common File GPC
  16: Convert ascii rake to Common File rake CGF
  17: Convert Pegsus 4.0 files to Common File

  Enter the number from one of the above requests
11

   PLOT3D file type menu

   0: Main menu
   1: Convert a PLOT3D Grid     (.x) file to CFS.
   2: Convert a PLOT3D Solution (.q) file to CFS.

  Enter the number from one of the above requests
1

   PLOT3D Number of Grids menu

   0: Main menu
   1: PLOT3D Single zone format.
   2: PLOT3D Multi  zone format.

  Enter the number from one of the above requests
2

   PLOT3D Zone dimension menu
```

```
   0: Main menu
   1: PLOT3D 2d zone format.
   2: PLOT3D 3d zone format.

  Enter the number from one of the above requests
1

   PLOT3D Format menu

   0: Main menu
   1: PLOT3D Formatted   (ASCII).
   2: PLOT3D Unformatted (sequential binary).
   3: PLOT3D Binary (c binary).

  Enter the number from one of the above requests
2

   PLOT3D Iblank menu

   0: Main menu
   1: PLOT3D grid with    IBLANK format.
   2: PLOT3D grid without IBLANK format.

  Enter the number from one of the above requests
2

   PLOT3D Precision menu

   0: Main menu
   1: PLOT3D Single precision format.
   2: PLOT3D Double precision format.

  Enter the number from one of the above requests
1

   PLOT3D INTOUT menu

   0: Main menu
   1: No INTOUT/XINTOUT file.
   2: INTOUT file.
   3: XINTOUT file.

  Enter the number from one of the above requests
1
  Enter PLOT3D .x file to convert with suffix
case4.xyz
  Enter output Common File name with suffix
case4.cgd
  Global maximums set as follows:
  mimax 33
```

```
 mjmax 11
 mkmax 1
 mpts: 363
 recl: 726
Processing zone ZONE   1
Writing mesh data
Processing zone ZONE   2
Writing mesh data
Processing zone ZONE   3
Writing mesh data

  0: Exit program
  2: Import   a Common File
  3: Compress a Common File
  4: Break Common File into multiple transfer files
  5: Combine multiple transfer files into Common File
  6: Append one Common File to another
  7: Convert Common File binary to a text file
  8: Convert Common File text to a binary file
 11: Convert PLOT3D/Pegsus file to Common File
 12: Convert GASP     file to Common File
 13: Convert OVERFLOW file to Common File
 14: Convert Common File to OVERFLOW file
 15: Convert CFPOST GPU file to Common File GPC
 16: Convert ascii rake to Common File rake CGF
 17: Convert Pegsus 4.0 files to Common File

 Enter the number from one of the above requests
0
```

2.3 Define the Input

The next step is to define the input data. Input is required to specify the flow and initial conditions, the boundary conditions, and various parameters controlling the physical and numerical models to be used when running the code.

The primary file controlling how the WIND program is executed is the *Input Data* (*.dat*) file. With many CFD codes the input data are specified using Fortran namelist and/or formatted input. With WIND, the input is specified using descriptive keywords. The formatting rules for the *.dat* file are described in Section 7.1.

This section is intended as an introduction to some of the more commonly-used keywords. After reading the information presented here, a new user should supplement it with the detailed information in Section 10. For many cases, the default values for the various keyword options are acceptable, but users should become familiar with all of the options for the most effective use of the WIND code.

2.3.1 Descriptive Header and Comments

The first three lines of the file are reserved for geometry, flow condition, and arbitrary titles, respectively. Each of these titles may be up to 64 characters long.

Comment lines, beginning with a /, may be placed anywhere in the file after the first three lines. The readability of the *.dat* file may be improved significantly through the liberal use of comments — for example, to separate logical sections of the data file like boundary conditions, numerical operators, convergence monitoring parameters, etc.

2.3.2 Boundary Conditions

With most CFD codes, boundary conditions are completely specified in the input data file. With WIND, however, setting boundary conditions is a two-step process, defining first the *type* of boundary, and then any *values* that are needed.

The first step is to label each boundary of each zone with the *type* of boundary condition to use, such as "viscous wall," "outflow," or "coupled." This is done using the GMAN pre-processing code, and the information is stored in the Common Grid (*.cgd*) file. Details on the boundary condition types available for use with WIND are in Section 5 and in the *GMAN User's Guide*.

Boundary condition types may be specified for all or part of a boundary, allowing multiple boundary conditions on a single boundary.

"Coupled" zonal interface boundaries do not have to be explicitly labeled by the user. GMAN can automatically examine the grid to find them and determine the zones involved, compute the geometric interpolation factors, and store the information in the *.cgd* file. GMAN is also used to cut holes and generate interpolation coefficients for overlapping ("chimera") boundaries. The process is currently not completely automated for chimera boundaries.

The second step is to define any *values* needed for a particular boundary condition, such as an exit pressure or a bleed rate. This information is specified in the *.dat* file. Keywords are available to specify conditions at inflow boundaries (`ARBITRARY INFLOW`), at outflow boundaries (`COMPRESSOR FACE`, `DOWNSTREAM MACH`, `DOWNSTREAM PRESSURE`, `MASS FLOW`), along solid walls (`MOVING WALL`, `TTSPEC`, `WALL TEMPERATURE`), in bleed and blowing regions (`BLEED`, `BLOW`), and across actuators and screens (`ACTUATOR | SCREEN`).

2.3.3 Flow and Initial Conditions

The flow conditions (Mach number, and static or total pressure and temperature, plus the angles of attack and yaw) are specified using the `FREESTREAM` keyword. These conditions, along with a reference length based on the units used in the *.cgd* file, are used as the reference conditions and determine the Reynolds number.

The usual procedure with WIND is to start a new problem by setting the initial conditions at each grid point equal to the values specified using the `FREESTREAM` keyword. Other keywords allow different values to be used in different zones (`ARBITRARY INFLOW`), a boundary layer to be added along a specified surface in a zone (`BL_INIT`), and reinitialization of the flow in specified zones after a restart (`REINITIALIZE`). Previously-run, partially-converged cases will normally be restarted using the current solution as initial conditions (`RESTART`). More information about flowfield initialization may be found in Section 3.6.

2.3.4 Physical Model Controls

Dimensionality. WIND may be used for three-dimensional, two-dimensional, quasi-three-dimensional, or axisymmetric configurations. Internally, the WIND code treats the grid as three-dimen-

sional, with indices i, j, and k. Two-dimensional cases simply have $k_{max} = 1$, with the i-j grid lying in a non-zero, constant z plane. The effect of area variation in an otherwise two-dimensional configuration may be modeled using WIND's quasi-three-dimensional capability, which is activated by setting the z-coordinate equal to the "width" of the geometry at each grid point. Axisymmetric configurations are modeled using a two-dimensional grid in conjunction with the AXISYMMETRIC keyword. More details may be found in Section 3.1.

Flow Equations. The default equations solved by the WIND code are the full Reynolds-averaged Navier-Stokes equations. Keywords are available to solve the Euler equations (TURBULENCE), parabolized Navier-Stokes equations (MARCHING), or the thin-layer Navier-Stokes equations (TSL). See Section 4.2.2 for more information about the thin-layer option.

Turbulence Model. A variety of turbulence models are available in the WIND code through the TURBULENCE keyword. These include the Baldwin-Lomax, Cebeci-Smith, and P. D. Thomas algebraic models; the Spalart-Allmaras and Baldwin-Barth one-equation models; and the Chien k-ϵ and Menter Shear Stress Transport (SST) two-equation models.[5] A laminar flow option is also available using the TURBULENCE keyword, and a laminar-turbulent transition region may be modeled using the TTSPEC keyword. See Section 3.3 for more information.

Gas Model and Chemistry. A variety of gas models are available in WIND to complete the equation set. The fluid may be treated as a thermally and calorically perfect gas, a thermally perfect gas (frozen chemistry), equilibrium air, or a mixture undergoing a finite rate chemical reaction. For a thermally and calorically perfect gas, the values of γ, the laminar and turbulent Prandtl numbers, and the gas constant may be set using the GAS keyword. Real gas chemistry models are selected using the CHEMISTRY keyword. Several different chemistry packages are available in the form of files containing thermodynamic data, finite rate coefficients, and transport property data, described in Section 7.10.

2.3.5 Numerical Model Controls

Time Stepping. In WIND, the number of iterations or time steps to perform in a given run is defined in terms of *cycles* and *iterations per cycle*. An iteration advances the solution one time step. A cycle consists of a solution pass over all the zones. Zone coupling, the process whereby WIND exchanges flowfield information between zones, only occurs at the end of each cycle. The Common Flow (*.cfl*) file is also updated only at the end of each cycle. The number of cycles to be performed is set using the CYCLES keyword, and the number of iterations per cycle, which may vary from zone to zone, is set using the ITERATIONS keyword. The default is five iterations per cycle.

The time step size is controlled by the CFL# keyword. By default, local time stepping is used, so that the solution advances at a different rate at each grid point. For unsteady problems, of course, the same time step size should be used throughout the flowfield, and an option is available with the CFL# keyword for this purpose. A Runge-Kutta time step formulation is also available, using the STAGES keyword, and may be used for both steady and unsteady flows.

See Section 4.1 for more information about cycles and iterations, and Section 4.5 for more about time step options.

[5] It should be noted that the algebraic models are older and infrequently used. They may not work as well as in some other codes, and there may be bugs in their implementation.

Implicit Operator. The IMPLICIT keyword allows a variety of implicit operators to be specified, including point Jacobi, Gauss-Seidel, and MacCormack modified approximate factorization. Also available are options to: (1) turn off the implicit operator completely, resulting in an explicit calculation; (2) use a scalar (diagonalized) implicit operator; or (3) use a full block implicit operator. For these last three options, a different implicit operator may be specified for each computational direction.

The default is to use the full block operator in viscous directions, and the scalar (diagonalized) operator in inviscid directions.

The IMPLICIT BOUNDARY keyword may be used to specify that implicit boundary conditions are to be used on "wall" boundaries. This should improve stability when the CFL number is above about 1.3.

Explicit Operator. Through use of the RHS keyword, a wide variety of explicit operators are available for evaluation of the first-derivative terms on the right-hand side. These include a central difference scheme, the upwind Coakley, Roe, Van Leer, and HLLE schemes, and modified versions of the Roe, Van Leer, and HLLE schemes for stretched grids. Depending on the type of scheme used, the accuracy may be specified as anywhere from first to fifth order. The default is Roe's second-order upwind-biased flux-difference splitting algorithm, modified for stretched grids.

Damping Schemes. Various smoothing options are available in WIND to dampen instabilities that may occur under certain conditions. These include second- and fourth-order explicit smoothing (SMOOTHING, BOUNDARY-DAMP), and total variation diminishing (TVD) flux limiting for some of the explicit operators (TVD, BOUNDARY TVD). More details on the various damping options are in Section 4.3.

Convergence Acceleration. The ACCELERATE keyword may be used, in conjunction with the SMOOTHING and CFL# keywords, to increase the time step near the beginning of a calculation, in order to more quickly get through the start-up transients that may occur in the first few hundred iterations.

A grid sequencing capability is also available, using the SEQUENCE keyword, that may help speed convergence. With this option, grid points are removed from selected regions of the flowfield, resulting in a coarse-grid solution which is obtained in a fraction of the time it would have taken for a full-grid solution. At the end of each run, the solution is interpolated back onto the original grid to aid in restarting the solution, and to provide a continuous flowfield for post-processing. The full grid should of course be used when the solution nears convergence.

See Section 4.6 for more information about the ACCELERATE option, and Section 4.2.1 for more about the grid sequencing capability.

Convergence Monitoring Parameters. The convergence criterion, in terms of the required value or reduction of the maximum residual, may be specified using the CONVERGE keyword. Integrated forces, moments, and/or mass flow may also be used to monitor convergence, by using the LOADS keyword to periodically compute and print these values for specified three-dimensional regions and/or two-dimensional areas of a computational surface.

For unsteady flow problems, where the time step is being specified in seconds and is the same throughout the flow field, a time history tracking capability is also available using the HISTORY

keyword. Selected flow variables may be computed at specified grid points, and written to a separate *Time History* (*.cth*) file.

More information about monitoring convergence is presented in Section 2.5 of this tutorial, and in Section 6.

2.3.6 Test Case 4 Input

Boundary Condition Types. *GMAN* was used in graphical mode to set the boundary condition types and store the information in the *.cgd* file. The interface boundaries between the three zones were automatically identified, and the geometric interpolation factors were computed and stored in the *.cgd* file. The inflow boundary ($i = 1$ in zones 1 and 2) was labeled as "arbitrary inflow," the outflow boundary ($i = i_{max}$ in zone 3) was labeled as "outflow," and the top and bottom boundaries ($j = 1$ in zones 1 and 3, and $j = j_{max}$ in zones 2 and 3) were labeled as "inviscid wall."

The first step, obviously, is to start GMAN.

```
gman

                              ***** gman *****

          Select the desired version from the following list.

0) END
1) gman_pre optimized version

Enter number of executable.......[1]:

 ************************************************************************
 * Warning: This software contains technical data whose export is       *
 * restricted by the Arms Export Control Act (Title 22, U.S.C., Sec 2751,   *
 * et seq.) or Executive Order 12470. Violation of these export-control     *
 * laws is subject to severe criminal penalties. Dissemination of this      *
 * software is controlled under DoD Directive 5230.25 and AFI 61-204.       *
 ************************************************************************
           GMANPRE - Version 6.153 (last changed 2002/11/05 20:30:13)

                         Creating journal file 'gman.jou'.

  Enter SWITCH or GRAPHICS to change to graphics mode.

 GMAN:
```

At this point, you may enter commands individually at the `GMAN:` prompt. Or, you could enter `SWITCH` or `GRAPHICS` to enter graphics mode.

The rest of this section describes in detail the use of GMAN for the tutorial test case. The graphics mode steps are on the left, with the Main Menu steps left-aligned and the Menu Options indented. Most of these are accomplished in GMAN by clicking on the listed menu item using the left mouse button. A few require entering text in the prompt area at the bottom of the screen. (See the "Graphical User Interface Basics" section of the *GMAN User's Guide* for a description of the various sections of the GMAN screen layout.)

The command line equivalents are shown on the right. Note that, in general, several graphics mode steps become consolidated into a single command.

We first need to tell GMAN the name of the file containing the grid.

Graphics Mode	**Command Line Mode**
`Common File` `  enter case4.cgd`	`file case4.cgd`

Next, we use GMAN's automated procedure to define zonal coupling information.

Graphics Mode	**Command Line Mode**

`BOUNDARY COND.`
`AUTO COUPLE`
`  RUN AUTO COUP`
	`automatic couple face zone all`

Next, for zone 1, we define the inflow and lower wall boundaries.

Graphics Mode	**Command Line Mode**
`PICK ZONE/BNDY` `  1:` (*from Zone List*)	`zone 1`
`  I1`	`boundary i1`
`MODIFY BNDY`	`arbitrary inflow`
`CHANGE ALL` `  ARBITRARY INFLO`	
`BOUNDARY COND.` `  YES - UPDATE FILE`	`update`
`PICK ZONE/BNDY` `  J1`	`boundary j1`
`MODIFY BNDY`	`inviscid wall`
`CHANGE ALL` `  INVISCID WALL`	
`BOUNDARY COND.` `  YES - UPDATE FILE`	`update`

For zone 2, we define the inflow and upper wall boundaries.

Graphics Mode	Command Line Mode
PICK ZONE/BNDY	zone 2
2: (*from Zone List*)	
I1	boundary i1
MODIFY BNDY	arbitrary inflow
CHANGE ALL	
ARBITRARY INFLO	
BOUNDARY COND.	update
YES - UPDATE FILE	
PICK ZONE/BNDY	
JMAX	boundary jmax
MODIFY BNDY	inviscid wall
CHANGE ALL	
INVISCID WALL	
BOUNDARY COND.	update
YES - UPDATE FILE	

And for zone 3, we define the outflow and both wall boundaries.

Graphics Mode	Command Line Mode
PICK ZONE/BNDY	zone 3
3: (*from Zone List*)	
IMAX	boundary imax
MODIFY BNDY	outflow
CHANGE ALL	
OUTFLOW	
BOUNDARY COND.	update
YES - UPDATE FILE	
PICK ZONE/BNDY	
J1	boundary j1
MODIFY BNDY	inviscid wall
CHANGE ALL	
INVISCID WALL	
BOUNDARY COND.	update
YES - UPDATE FILE	
PICK ZONE/BNDY	
JMAX	boundary jmax
MODIFY BNDY	inviscid wall
CHANGE ALL	
INVISCID WALL	
BOUNDARY COND.	update
YES - UPDATE FILE	

Finally, it's a good idea to check the boundary conditions to make sure all is OK.

Graphics Mode	**Command Line Mode**
TOP	
CHECK	zone all
CHECK BOUNDARY	check boundary
PICK ZONE	
ALL (*from Zone List*)	
RUN BNDY CHKS	

After hitting the Enter key to return to GRAPHICS mode, we can quit GMAN.

Graphics Mode	**Command Line Mode**
END	exit
YES - TERMINATE	

Input Data (*.dat*) File. The Input Data File for Test Case 4, named *case4.dat*, is listed below. The explanatory notes, in italics, are not part of the file.

```
WIND test case 4, 2-D, 3 zones          Titles
Subsonic internal flow
Run 1

/ Inlet conditions
Freestream total 0.78 15.0 600.0 0. 0.  Inlet M, pt (psi), Tt (°R), α, β

/ Boundary conditions
Downstream pressure 14.13 zone 3        Exit p (psi)

/ Numerics
Cycles 500                              Run 500 cycles
Iterations 5 Print frequency 5          5 iterations/cycle; print every 5th

/ Viscous terms
Turbulence euler                        Solve inviscid equations

/ Convergence data
Loads                                   Compute ṁ every 5 iterations at:
   print planes frequency 5
   zone 1
      surface i 1 mass                      zone 1 entrance
   zone 2
      surface i 1 mass                      zone 2 entrance
   zone 3
      surface i 1    mass                   zone 3 entrance
      surface i last mass                   zone 3 exit
Endloads
```

2.4 Run the Code

2.4.1 The *wind* Script

WIND is invoked using a Unix script, called *wind*, which prompts for the executable to be used (since production and beta versions of WIND may both be available on a system), the names of the various input and output files (which should be entered *without* the three-letter suffix), and for the queue in which the job is to run. If a *multi-processing control* (*.mpc*) file is present with the same base name as the *.dat* file (see Section 2.4.2), it also issues a prompt to verify that the job is to be run in parallel mode. The script then links the files to the appropriate Fortran units, and either starts WIND interactively or submits the job to the specified "at" or "batch" queue. Details on the *wind* script are in Section 8.1.

There are a couple of very convenient features built into the *wind* script. The first allows a run to be stopped at (or more exactly, shortly after) a pre-determined time through the use of an *NDSTOP* file. This is useful when an overnight run must be stopped before morning, when the workstations being used will be needed for interactive work. The second allows the user to break a long run into "sub-runs," by writing a script called *wind_post* containing tasks to perform between each run. This is useful, for example, when the complete solution is to be saved at various time intervals in an unsteady problem. Details on the use of the *NDSTOP* file and the *wind_post* script are in Section 7.8 and Section 8.2, respectively.

2.4.2 Parallel Operation

When WIND is run in parallel mode, multiple systems connected via a network work together as though they were a single computer. These systems are typically workstation class machines and need not be all from the same vendor.

A master-worker approach is used. Grid zones are distributed from the master system to the worker systems for processing. (Note that the master may also be a worker.) Each zone is solved in parallel with other zones on other systems. The systems exchange boundary information at the end of each solution cycle to propagate information throughout the flowfield. If there are fewer workers than zones, a worker will be assigned another zone when it finishes its current assignment.

The user specifies the names of the participating worker systems via a *multi-processing control* (*.mpc*) file, which must have the same base name as the *.dat* file. The user must of course have accounts on the master and worker systems, and the master must have remote shell access to each of the workers, via a *hosts.equiv* or *.rhosts* file.[6] This is all that is required to utilize the distributed parallel processing capability of WIND. The PVM software needed for parallel operation, and the WIND code itself, will be copied from the master to temporary directories on the workers.

Additional details about running WIND in parallel mode may be found in Section 9.

2.4.3 Running Test Case 4

To run Test Case 4, simply issue the *wind* command, and respond to the prompts as appropriate. The following terminal session shows how the case was run as a serial batch job on a Unix workstation. Lines in slanted type were typed by the user.

```
% wind -runinplace
```

[6] If *ssh* (secure shell) is being used, the corresponding files are *shosts.equiv* and *.shosts*.

```
Running command line version of WIND.


                  ***** WIND Run Script *****


Current wind settings are:

--Wind test mode set to NODEBUG
--Wind debugger set to DEFAULT
--Wind run que set to PROMPT
--Wind run in place mode is set to YES
--Wind multi-processor mode set to NO
--Wind run directory set to PROMPT
--Wind bin directory set to /usr2/wind/wind

                Select the desired version

  0: Exit wind
  1: Wind Version 3.0
  2: Wind Version 4.0
  3: Wind Version 5.0

Enter number or name of executable.......[3]:
Basic input data................(*.dat): case4
Output data.............(*.lis,<CR>=case4):
Mesh file...............(*.cgd,<CR>=case4):
Flow data file..........(*.cfl,<CR>=case4):

************************************************************
case4.cfl does not exist, a fresh start will be performed.
************************************************************


Enter a queue number from the following list or <CR> for default:
  1: REAL (interactive)
  2: AT_QUE
  3: QSUB_QUE
  4: QSUB_PBS_QUE

Queue name.....................(<CR> for 1): 2


Enter at que (a,b..)................(<CR> for default):
Deferred start time..............(hhmm [day],<CR>=now):
Set stop flag at............(hhmm [day],<CR>=don't set):

Version..............: /usr2/wind/wind/SGIMP6.5/R12000/wind5.exe
Input file name.......: case4.dat
Wind output to........: case4.lis
Grid file name........: case4.cgd
```

```
Flow file name........: case4.cfl
Job run que type is...: AT_QUE

Press <cr> to submit job, another key (except space) and <cr> to abort:
       warning: commands will be executed using /bin/sh
job 1090343885.b at Tue Jul 20 13:18:05 2004
All done, have a nice day!!
```

There are several points to note from this terminal session.

- This case was run using the *-runinplace* option to the *wind* script, which means that the WIND code will be run in the current directory, and that output files will be written in the current directory.

  The default is to run in a different (i.e., remote) directory, and is intended primarily for use with NFS-mounted home directories. In that case, it's faster to write the output files into a scratch directory on the system used to run the WIND code, rather than into the NFS-mounted home directory. The output files are automatically copied to the current directory at the end of the job.

  Note that the terminology here is unfortunately a bit confusing. With an NFS-mounted home directory, running remotely really means running on a system different from the one the home directory is on. The "remote" system may actually be the local system originally logged onto.

  If this case were run without the *-runinplace* option, the user would be prompted to enter the root name of the remote run directory, as follows:

  ```
  # Note the remote directory is assumed to exist on remote host #
  Enter the remote run root directory...(<CR> for /tmp):
  ```

  The full name of the remote run directory will be *rootname/userid/basename.scr*, where *rootname* is your response to the above prompt, *userid* is your userid, and *basename* is the base name of your *.dat* file. The default of */tmp* for the root name implies that, generally, the "remote" system is actually the one the user logged into. It also means that, if you aren't using NFS-mounted home directories, and you forget to add the *-runinplace* option, no real harm is done. The output will be created under */tmp*, then copied to the current directory when the job finishes.

- The default for the base name of the *.lis*, *.cgd*, and *.cfl* files is the same as that entered for the *.dat* file, and the three-letter suffixes *should not* be entered.

- A "fresh start" is being done for this case, since the *.cfl* file does not exist. If the *.cfl* file exists, WIND will automatically restart from the existing flow field.

- To run the job interactively, choose the `REAL` queue. This is intended mostly for debugging short runs, or checking that a case will successfully start. The user will be asked if the output should be written to the screen instead of the *.lis* file.

  The `AT_QUE` queue is used to run a batch job using the Unix *at* or *batch* command. If the response to the `Deferred start time` prompt is defaulted, the job will be started immediately using the *batch* command. Any other response will result in the *at* command being used to start the job at the specified time.[7]

  The `QSUB` queue choices shown above are intended for use on systems with NQS (Network Queueing System) software installed. The user will be prompted for the necessary NQS queue name, start time, etc. The `QSUB` choices will not appear on systems without a *qsub* command.

[7] *Do not* type `now` in response to the prompt. That will result in an "at" job being submitted, and a "too late" error.

- As noted earlier, if a multi-processing control (*.mpc*) file is present, the user is also asked to verify that the job is to be run in parallel mode, as follows:

```
A multiprocessing control file exists....
Do you want to run in multi-processor mode (y/n,<CR>=y):
```

2.5 Monitor Convergence

For complex real-world applications, it is generally not feasible to expect a converged solution in a single run. The times required to achieve convergence are generally too long and problems may occur which could corrupt the solution. Thus, executing the code several times and restarting from the previous solution is often the best approach. If problems do occur, the input parameters can be adjusted without starting from scratch.

Monitoring and properly assessing convergence levels during a WIND run (as well as examining the flowfield itself, as discussed in Section 2.6) are thus crucial in obtaining meaningful, useful results. WIND users may track convergence by following residuals and/or integrated forces, moments, and mass flow via the `LOADS` keyword. For engineering applications, the recommended convergence monitoring method is the tracking of integrated quantities of interest. For example, if a wing/body geometry is being modeled to determine drag, the integrated drag should be monitored and some reasonable bounds on drag oscillations should be used as the convergence criterion.

The solution residuals are included in the *List Output* (*.lis*) file. For each zone, WIND prints the zone number, cycle number, location of the maximum residual, equation number for which the maximum residual occurred, the value of the maximum residual, and the L_2-norm of all the residuals for all the equations over all the points in the zone. By default, the residuals are printed each iteration. The output interval may be changed, however, using the `CYCLES` and `ITERATIONS` keywords.

The integrated parameters that are chosen in the Input Data file via the `LOADS` keyword will also be listed in the *.lis* file. The integration may be done over a number of specified three-dimensional regions and/or two-dimensional areas of a computational surface.

For unsteady flow problems, where a constant time step is being specified and is the same throughout the flow field, a time history tracking capability may be used. Computed values of selected variables at specified grid points may be periodically written to a separate *Time History* (*.cth*) file. This capability is activated using the `HISTORY` keyword.

A utility included with the WIND code called *resplt* can be used to extract the residuals and/or integrated quantities from the *.lis* file, and create a GENPLOT file[8] for post-processing.

An analogous utility called *thplt* can be used for the values stored in the *.cth* file.

Additional information about the various methods for monitoring convergence is presented in Section 6.

2.5.1 Test Case 4 Convergence

resplt was used to extract the maximum residual and the L_2-norm of the residuals from the List Output (*.lis*) file and create GENPLOT files. As an example of the use of *resplt*, the following terminal session shows how a GENPLOT file containing the maximum residual was created. A

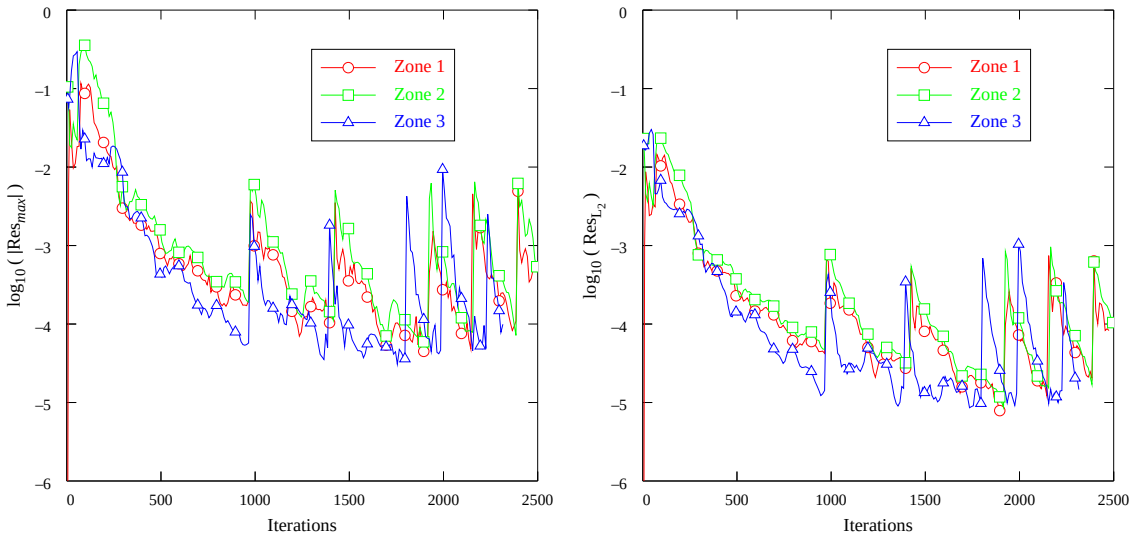
[8]The format of a GENPLOT file is described in the *CFPOST User's Guide*.

GENPLOT file containing the L_2-norm of the residual was created similarly, using selection "2". Lines in slanted type were typed by the user.

```
% resplt

                          ***** resplt *****

          Select the desired version from the following list.

0) END
1) resplt optimized version

Single program automatically selected.

 resplt - Version 1.8 (last changed 2004/04/09 14:44:52)
 ************************************************************************
 * Warning: This software contains technical data whose export is       *
 * restricted by the Arms Export Control Act (Title 22, U.S.C., Sec 2751,   *
 * et seq.) or Executive Order 12470. Violation of these export-control     *
 * laws is subject to severe criminal penalties. Dissemination of this      *
 * software is controlled under DoD Directive 5230.25 and AFI 61-204.       *
 ************************************************************************
  Enter full name of output list file:
case4.lis

Exit                0
Select Zone(s)     91
Select Frequency   92
Select average mode 99
Confined Outflow
  Mass Flow Ratio  15
  Back Pressure    16
  Average p0       93
Residuals     Big   L2       Integ. Planes Zone Grand
  NS            1    2         Force    11    5    8
  k-e           3    4         Lift     17   18   19
  B-B          20   21         Moment   12    6    9
  S-A          22   23         Momentum 13    7   10
  SST          24   25         Mass     14   26    -
NEWTON NS      51   52         Heat Flx 54   55    -
Time History       53

Enter Selection
1
  Reading residual data...

  1466 data points read from list file.


  Sorting residual data...
  Sorting complete.
```

```
  Enter FULL name of genplot file:
chist.big.gen

Exit                 0
Select Zone(s)      91
Select Frequency    92
Select average mode 99
Confined Outflow
  Mass Flow Ratio   15
  Back Pressure     16
  Average p0        93
Residuals     Big   L2       Integ. Planes Zone Grand
  NS            1    2         Force    11    5    8
  k-e           3    4         Lift     17   18   19
  B-B          20   21         Moment   12    6    9
  S-A          22   23         Momentum 13    7   10
  SST          24   25         Mass     14   26    -
NEWTON NS      51   52         Heat Flx 54   55    -
Time History         53

Enter Selection
0
```

The convergence history for Test Case 4, in terms of the maximum residual and the L_2 norm of the residual, is shown in Figure 3.



(a) Maximum residual

(b) L_2 norm of the residual

Figure 3: Test case convergence history in terms of residuals

For this case, the residuals decrease about three orders of magnitude in the first 1000 or so

iterations, then oscillate about a relatively constant value for the remainder of the iterations. This behavior is not at all uncommon. As noted above, it's usually more meaningful to use a physical quantity of interest when monitoring convergence. For this test case, one of the physical quantities of interest is the mass flow rate.

The Input Data file for this case specified that the integrated mass flux was to be computed at the entrances to all three zones, and at the exit of zone 3. GENPLOT files containing these integrated values were created using *resplt*, as illustrated above, using selection "14".

The time history for these parameters is shown in Figure 4, where the computed mass flux is plotted as a function of iteration number at the entrances to all three zones, and at the exit of the duct.

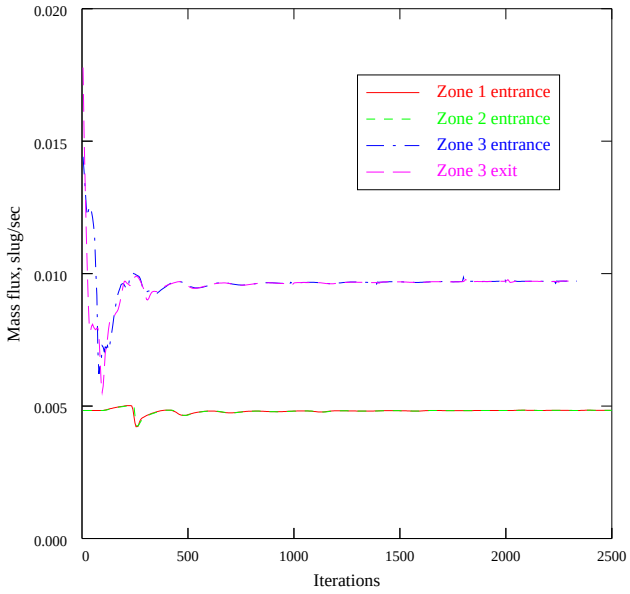


Figure 4: Test case convergence history in terms of mass flux

Based on the mass flux results, this case appears to converge within about 750–1000 iterations. Because this is a simple 2-D inviscid flow, with a coarse mesh, it converged quickly in a single run. The various convergence parameters were thus examined only at the end of the run. When running a more realistic, real-world configuration, convergence parameters like those shown in Figure 3 and Figure 4 should be examined at the end of each run. A more complete determination of convergence would also include examination of other physical quantities, such as the pressure distribution along the duct.

2.6 Examine the Results

Of course, the purpose of the solution process is to determine the features of the flow which can help answer the questions that drove the decision to perform the simulation in the first place. And, as indicated in the previous section, it's important to periodically examine the computed results during a run to help assess convergence and detect numerical problems that might be corrected by adjusting the input.

There are two types of information that can be extracted from the flow simulation — specific

quantitative data and qualitative patterns. The first type includes things like pressure distributions, drag, and total flow rate. The second type includes, for example, 2-D slices or full 3-D visualization of pressure contours, streamlines, or isothermal surfaces.

2.6.1 Using CFPOST

All flowfield results computed by WIND, including the mean flow variables, turbulence model variables, and chemistry variables, are written into a Common File called a *Common Flow* (*.cfl*) file.[9] The CFPOST utility, included with the WIND distribution, is a post-processing tool for examining the contents of the *.cfl* file.

With CFPOST a wide variety of variables and integrated values may be computed. Listings of quantitative results may be sent to the screen or to a file. PLOT3D files may be created for other plotting packages and post-processors to use in displaying qualitative results. CFPOST can also be used to create *x*-*y*, contour, and vector plots directly, with PostScript output. Commands are available to precisely specify the information of interest, the domain of interest, and the units in which the results are to be presented. More details may be found in the *CFPOST User's Guide*.

2.6.2 Test Case 4 Results

The desired results from this calculation were the mass flow rate and the static pressure distribution. The mass flow rate is available in the List Output (*.lis*) file, as output generated via the `LOADS` keyword, and also from the GENPLOT files created while monitoring convergence. The result was 9.7×10^{-3} slug/sec.

To examine the static pressure distribution, CFPOST was first used to create a PLOT3D q file called *case4.q* from the Common Flow file *case4.cfl*, as follows:[10]

```
% cfpost

                          ***** cfpost *****

          Select the desired version from the following list.

0) END
1) cfpost_pre optimized version

Enter number of executable.......[1]:

           CFPOST - Version 3.163 (last changed 2002/11/05 20:28:28)
 ************************************************************************
 * Warning: This software contains technical data whose export is       *
 * restricted by the Arms Export Control Act (Title 22, U.S.C., Sec 2751,   *
 * et seq.) or Executive Order 12470. Violation of these export-control     *
 * laws is subject to severe criminal penalties. Dissemination of this      *
 * software is controlled under DoD Directive 5230.25 and AFI 61-204.       *
 ************************************************************************
```

[9] As noted previously, WIND also supports the use of CGNS files for the grid and flow solution, using the `CGNSBASE` keyword.

[10] CFPOST can also be used to create a PLOT3D xyz file, but as noted in Section 2.2.3, an xyz file for this configuration was created during the mesh generation process.

```
 CFPOST> solution case4.cfl
1WIND test case 4, 2-D, 3 zones
 Subsonic internal flow

 Freestream conditions (* indicates calculated)

 Mach number                   0.780
 Angle of attack               0.000 degrees
 Yaw angle                     0.000 degrees
 Gamma                         1.400
 Gas constant (R)            286.959 m2/s2-K         1716.000 ft2/s2-R
 Static pressure           69194.1     N/m2          10.0357     lbf/in2
 Static temperature        297.173     K             534.912     R
 Static density*          0.811408     kg/m3        0.157439E-02 slug/ft3
 Total pressure*           103421.     N/m2          15.0000     lbf/in2
 Total temperature*        333.333     K             600.000     R
 Total density*            1.08121     kg/m3        0.209790E-02 slug/ft3
 Speed of sound            345.524     m/s           1133.61     ft/s
 Velocity                  269.509     m/s           884.216     ft/s
 Dynamic pressure          29468.4     N/m2          4.27402     lbf/in2
 Laminar Viscosity        0.183722E-04 kg/m-s        0.383710E-06 slug/ft-s
 Reynolds Number*         0.119029E+08 1/m           0.362800E+07 1/ft
 Enthalpy*                 298468.     m2/s2         0.321268E+07 ft2/s2
 Stagnation Enthalpy*      334785.     m2/s2         0.360360E+07 ft2/s2
 CFPOST> subset i all j all k all
 CFPOST> zone 1
 CFPOST> zone 2
 CFPOST> zone 3
 CFPOST> plot3d q case4.q 2d mgrid unformatted
 Processing ZONE   1...
 Processing ZONE   2...
 Processing ZONE   3...
 CFPOST> quit
```

The PLOT3D xyz and q files were then used as input to other post-processing packages to examine the computed results. The computed static pressure distribution along the upper and lower walls is shown in Figure 5, and the static pressure field is shown in Figure 6.

2.7 Summary

The steps in the generalized solution process listed earlier may be expanded and restated specifically for the WIND code as follows:

- Gather information detailed enough to specify the problem within the required accuracy, including the geometry, flow conditions, and boundary conditions.
- Create a grid file using any convenient grid generation software, saving the file in PLOT3D xyz format.
- Convert the PLOT3D xyz file to a Common Grid (*.cgd*) file using *cfcnvt*.
- Store the boundary condition types and zonal connectivity data in the *.cgd* file using GMAN.

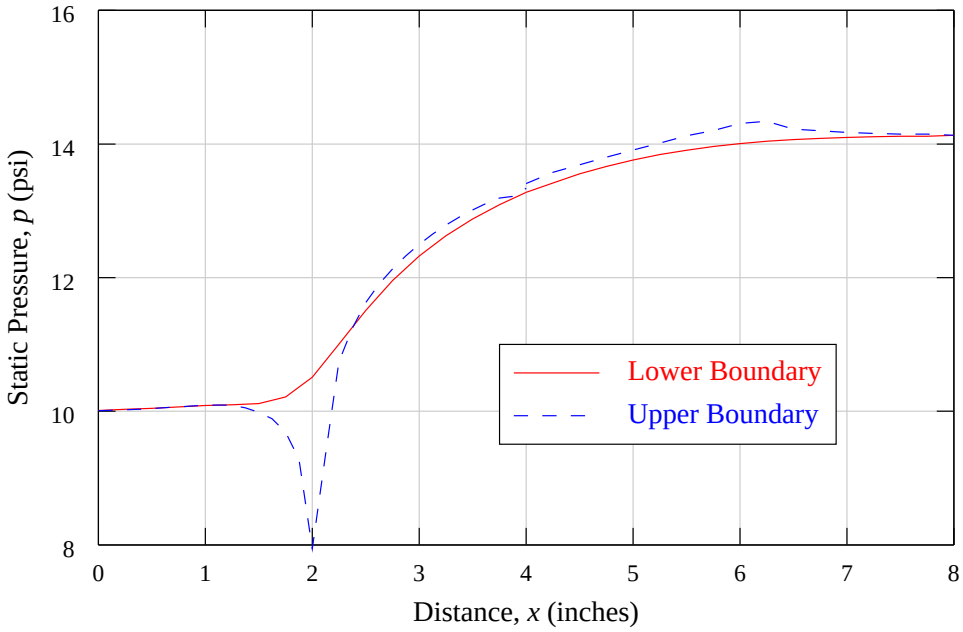


Figure 5: Test case static pressure distribution

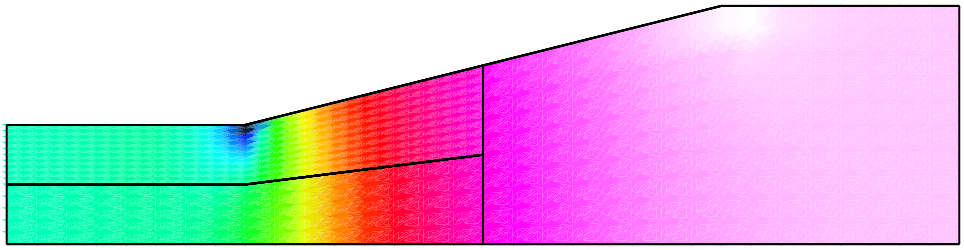


Figure 6: Test case static pressure field

- Prepare the Input Data (*.dat*) file, defining boundary condition values, initial conditions, program control parameters, and integrated parameters for monitoring convergence.
- For parallel execution, prepare the *multi-processing control* (*.mpc*) file.
- Run the WIND code using the *wind* script supplied with the code.
- Monitor convergence by examining the residuals and integrated values in the List Output (*.lis*) file. The utilities *resplt* and CFPOST may be helpful.
- Periodically examine the computed results in the Common Flow (*.cfl*) file using CFPOST, creating PLOT3D files for other post-processing packages if desired.

3 Geometry and Flow Physics Modeling

3.1 Symmetry Considerations

WIND may be used for axisymmetric, two-dimensional, or three-dimensional geometric configurations. Two-dimensional grids may be used not only for two-dimensional cases, but also for axisymmetric and area variation (quasi-three-dimensional) cases.

3.1.1 Three-Dimensional Cases

In three dimensions, each zone's computational mesh is comprised of six boundary faces and an interior grid, the points of which are identified by three indices, usually labeled (i, j, k). Boundary conditions for each of the six faces must be specified with GMAN before the grid may be used with WIND.

3.1.2 Two-Dimensional Cases

In two dimensions, the grid must be oriented such that the maximum k-index of the grid is 1. In other words, a two-dimensional grid is defined by four boundary faces and an interior grid labeled by i- and j-indices. The grid must also reside in a non-zero, constant z-coordinate plane. The actual value used will not affect solution convergence or flowfield features, but it will affect flux-related post-processing calculations such as mass flow. For this reason, a value of 1.0 is recommended. Boundary conditions for the four boundary lines must be specified in GMAN.

3.1.3 Area Variation (Quasi-3D) Cases

The effect of area variation on two-dimensional computational models may be computed by using WIND's "quasi-three-dimensional" capability, which is activated simply through changes in the z-coordinate. The value of the z-coordinate is the "width" of the field at each grid point; the complete grid therefore represents the "width" variation of the field as a function of x and y. The important quantity to model is the ratio of cross-sectional areas between two adjacent axial stations. This means that the z-coordinate may be scaled with no effect on the computed flowfield, but a simple translation of the z-coordinates will change the computed flowfield, because the cross-sectional area ratio will be different. As with two-dimensional calculations, the value of the z-coordinate will affect flux-related post-processing calculations.

3.1.4 Axisymmetric Cases

Axisymmetric configurations may be modeled using a two-dimensional grid generated at an arbitrary circumferential location on the geometry — e.g., the top centerline. Note that the grid should be generated on only one "side" of the configuration. Once again, the z-coordinate of the grid should be 1.0. The final step in using WIND's axisymmetric mode is the specification of the symmetry axis location and the circumferential sweep angle in the input data file. The circumferential sweep angle is the angle of the "pie shape" swept out by the grid about the symmetry axis. Although the value of the sweep angle will not affect the computed flowfield, it will affect flux-related post-processing calculations.

Keywords: AXISYMMETRIC

3.2 Euler and Navier-Stokes Equations

WIND may be used to solve the Euler equations or the Reynolds-averaged form of the Navier-Stokes equations (Bush, 1988). All heat transfer and stress tensor terms are retained, and the equations are modeled in full conservation form. The effects of turbulence may be modeled using a variety of algebraic, one-equation, and two-equation turbulence models. Modification of the effective heat transport coefficient due to turbulence is linked to the momentum diffusion coefficient by a turbulent Prandtl number, which is assumed to be constant.

The fluid may be treated as a thermally and calorically perfect gas, a thermally perfect gas, equilibrium air, or a mixture undergoing a finite rate chemical reaction. For an ideal gas, conventional values are given to the gas constant (R) and the ratio of specific heats (γ), or they may be specified. Effects of gravity (i.e., stratification) and rotation may also be included.

The equation set(s) to be solved must be specified in the input data file.

Keywords: `TURBULENCE`, `GRAVITY`, `ROTATE`

3.2.1 Freestream Conditions

Freestream flowfield conditions — Mach number, pressure, temperature, angle of attack, and angle of sideslip — must be specified in WIND's input data file. The Mach number must be greater than 0, and pressure and temperature may be specified as static or total values. These conditions are used to initialize the flowfield at the start of a run. For external flow problems, they are also are applied at all inflow, outflow, and freestream boundaries during the course of a flow solution. For this reason, the outermost grid boundary should be far enough away from the body such that the freestream assumption is valid at the boundary.

Keywords: `FREESTREAM`

3.2.2 Reynolds Number Considerations

WIND does not allow the direct specification of a Reynolds number; freestream conditions must be specified that are consistent with the desired value. Currently, WIND uses a reference length which it computes based on the unit system of the grid file. The code prints out the reference length (in inches) that it uses, so that you may compare the Reynolds number from the code with the desired value. The Reynolds number in WIND's output is actually the Reynolds number "per reference length."

Here are a couple of examples.

Case I: Grid is same size as model

Suppose we have the grid for a wind tunnel model, and we want to run it at $M_\infty = 0.7$ and a Reynolds number of 12.1 million (based on the model length). We arbitrarily choose a total temperature of 520 °R (static temperature of 473.6 °R). Knowing the temperature, we can calculate the speed of sound ($a = \sqrt{\gamma RT}$) and viscosity (from Sutherland's law). Using the Mach number, we

can calculate the freestream velocity. Thus, we have

$$\begin{aligned} M_\infty &= 0.7 \\ T_\infty &= 473.6\ ^\circ\text{R} \\ U_\infty &= 746.67\ \text{ft/sec} \\ \mu_\infty &= 3.474 \times 10^{-7}\ \text{lb}_\text{f}\text{-sec/ft}^2 \\ Re_L &= 12,100,000 \end{aligned}$$

We now have the Mach number and temperature for the WIND input data file, but we still need to calculate the freestream pressure. Using the definition of the Reynolds number (and the ideal gas law),

$$Re_L = \frac{\rho U L}{\mu} = \frac{P U L}{R T \mu}$$

or

$$P = \frac{R T \mu Re}{U L} = \frac{R T \mu Re_L}{U}$$

Note that Re_L is the Reynolds number per unit length of the physical model. For our example, if the model length is 10 inches,

$$\begin{aligned} Re_L &= 12,100,000/(10/12\ \text{ft}) \\ &= 14,520,000(1/\text{ft}) \end{aligned}$$

We can now calculate P:

$$\begin{aligned} P = \frac{R T \mu Re_L}{U} &= \frac{(1716)(473.6)(3.474 \times 10^{-7})(14.52 \times 10^6)}{746.67} \\ &= 5490.31\ \text{lb}_\text{f}/\text{ft}^2 = 38.13\ \text{psi} \end{aligned}$$

We would now like to check our input. If we run WIND with the Mach, temperature and pressure specified above, the code will print a Reynolds number and a reference length near the top of the list output file. *To obtain the desired Reynolds number, divide WIND's value of the Reynolds number by the output reference length and multiply by the model body length. This number may be compared with the desired model Reynolds number.*

Case II: Grid is scaled from model size

Let us now assume that we want to run the previous grid at flight conditions, but we want to keep our same old 10-inch grid. We simply need to multiply the pressure by a scale factor. The equation now becomes:

$$P = \frac{R T \mu Re_L S}{U}$$

where

$$S = \frac{\text{Full Model Size}}{\text{Grid Model Size}}$$

For example, if we want to run a 100-inch wing using our 10-inch grid, $S = 10$. If we want to run a flight Reynolds number of 26 million, we calculate P as:

$$\begin{aligned} P &= \frac{(1716.5)(473.6)(3.47 \times 10^{-7})(26 \times 10^6)(10)}{746.67} \\ &= 983113.0\ \text{lb}_\text{f}/\text{ft}^2 = 682.72\ \text{psi} \end{aligned}$$

3.2.3 Mass Flow in Two-Dimensional Calculations

One of the options available in WIND is the specification of mass flow boundary conditions for subsonic duct analyses. Actual or corrected mass flow may be specified at duct exits, as may back pressure. Regardless of the exit condition specified, WIND translates it into a uniform static pressure at the exit (which may or may not remain constant).

Within WIND and many post-processors, routines exist which integrate mass flow at desired computational planes. For 3D cases, the desired mass flow may be compared directly with the output from the integration routines:

$$\dot{m} = \int \rho u \, dA$$

However, for 2D calculations, the comparison is not so straightforward. There are three cases to consider.

Case I: 2D, Unit Depth

The first case involves running WIND on a truly two-dimensional grid of *unit depth* (z-coordinate is 1.0 everywhere). In this case, the input mass flow should be *per unit depth*. For example, let's say we want to run a 2D, unit depth model of a duct with a square exit. (If the exit were not square, this model would probably not be very good.) We would like a corrected mass flow of 500 $\mathrm{lb_m}$/sec, and our actual model exit depth is 10 inches. If the grid input units are inches, we should ask for a mass flow of 50 $\mathrm{lb_m}$/sec. If the grid input units are *not* inches, simply divide the actual mass flow by the z-coordinate value *in inches*.

Case II: 2D, Variable Width

When the z-coordinate is the width of the 2D grid, WIND adds in the area variation as a source to the 2D equations, making the analysis quasi-three-dimensional. In this case, the actual 3D mass flow should be specified in the input file. The integrated exit area will be (approximately, see Section 3.2.4 the real duct exit area, if the width has been specified correctly.

Case III: 2D, Axisymmetric

Axisymmetric runs require specification of the symmetry axis location and the circumferential angle subtended by the 2D grid. (This angle has no influence on the solution, but it does determine the area perpendicular to the grid.) The only reason this angle is an input parameter is so that you will know what the streamwise area is. In this case, the real exit geometry is circular, with a corresponding mass flow. The ratio of the input mass flow to the actual mass flow should equal the ratio of the input circumferential angle to 360. For example, if we are modeling a circular duct with a mass flow of 200 $\mathrm{lb_m}$/sec using an axisymmetric model in WIND, and if we specify a circumferential angle of 36 (1/10 of 360), we should specify a mass flow of 20 $\mathrm{lb_m}$/sec (1/10 of 200 $\mathrm{lb_m}$/sec).

3.2.4 Mass Flow and Grid Areas

When dealing with subsonic duct analyses, you should be aware that the duct area as represented by the grid may be slightly different from the real area of the geometry being modeled, especially for ducts modeled with quasi-polar grids.

The duct area represented by the computational grid is often smaller than the real duct area, which, when running near critical mass flow, may prematurely choke the flow in the CFD solution. If the duct is circular and is modeled with a quasi-polar grid, the area error may be estimated.

Suppose we are modeling a circular duct with a quasi-polar grid using $k_{max} = 33$ circumferential

points, each of which lie on the perimeter of the real duct at some streamwise station. If the circumferential points are evenly distributed, we may describe this topology as a circle which circumscribes a regular polygon of $k_{max} - 1 = 32$ sides. For a circle of radius R circumscribing a regular polygon of n sides, the area of the polygon is

$$A_p = \frac{1}{2} n R^2 \sin \frac{360}{n}$$

which means that the "grid area" is

$$A_g = \frac{1}{2}(k_{max} - 1) R^2 \sin \frac{360}{k_{max} - 1}$$

For our example, with $k_{max} = 33$, the grid area is 0.7% lower than the actual area.

There is no need to worry about this difference for most cases, but you should be aware of its possible effects.

3.2.5 Heat Transfer

At solid walls, WIND uses an adiabatic heat transfer boundary condition by default. A constant wall temperature may also be specified in the input data file. Through the use of the TTSPEC keyword, point-by-point wall temperature distributions may also be specifed on boundary surfaces. An auxiliary code, *tmptrn*, is used to create the wall temperature distribution, and write it into the common flow (*.cfl*) file.

The thermal conductivity is determined using a constant Prandtl number.

Keywords: WALL TEMPERATURE, TTSPEC

3.2.6 Viscosity

By default, WIND uses Sutherland's law to define laminar viscosity as a function of temperature. Keye's formula may be used in addition to Sutherland's law, and Wilke's law may be used to compute the laminar viscosity for multi-species flows.

Keywords: VISCOSITY

3.3 Turbulence Models

For turbulent calculations, Reynolds averaging assumptions are used to define a turbulent (eddy) viscosity, which is added to the laminar viscosity in the flow calculations. All the turbulence models available in WIND are coupled to the Navier-Stokes equations only through the turbulent viscosity. Several algebraic, one-equation, and two-equation turbulence models are available.

Note that a turbulence model (or inviscid or laminar flow) must be specified in the input data file. WIND will stop if you do not.

3.3.1 Algebraic Models

The algebraic turbulence models available in WIND are the Cebeci-Smith model, the Baldwin-Lomax model (Baldwin and Lomax, 1978), and the P. D. Thomas model (Thomas, 1979), which

adds a shear layer model to the Baldwin-Lomax model. After each iteration of the flow solver, these models compute the turbulent viscosity based on current flowfield quantities. Note that, because of their dependence on maxima and minima of flowfield variables, these models produce discontinuous turbulent viscosity distributions in the computed flowfield and require special numerical treatment at the juncture of two walls and on computational i-boundaries (the latter for historical reasons). The Baldwin-Lomax model is the most widely used algebraic turbulence model in WIND.

Keywords: TURBULENCE

3.3.2 One-Equation Models

Because of their efficiency and ability to produce continuous turbulent viscosity distributions, the one-equation turbulence models in WIND are the models of choice for many engineering applications. The one-equation models available in WIND are the Baldwin-Barth (Baldwin and Barth, 1990) and Spalart-Allmaras (Spalart and Allmaras, 1992) models.

For three-dimensional unsteady flows, the Spalart Detached Eddy Simulation (DES) model may be used.(Spalart, Jou, Strelets, and Allmaras, 1997; Shur, Spalart, Strelets, and Travin, 1999) This reduces to the standard Spalart-Allmaras model near viscous walls, where the grid is refined and has a large aspect ratio, but acts like a Large Eddy Simulation (LES) model away from the boundary, where the grid is coarser and has an aspect ratio of order one. It is intended to improve the results for unsteady and massively separated flows.

Keywords: TURBULENCE

3.3.3 Two-Equation Models

Two two-equation turbulence models are currently available in the WIND code — the Menter Shear Stress Transport (SST) model (Menter, 1993; Mani, Ladd, Cain, and Bush, 1997) and the low-Reynolds-number Chien k-ϵ model (Chien, 1982).

The Menter Shear Stress Transport (SST) model is a blend of k-ϵ and k-ω. The equations are cast in k-ω form. Near the boundary the k-ω model is used, and the k-ϵ model is used away from the wall and in shear layers. Freestream values of k and ω may be specified. Each equation is solved individually and an iterative method has been used on the left-hand side to reduce the factorization error. The model is robust, and may be more accurate in adverse pressure gradients than some of the other models in WIND.

For unsteady flows, a combined SST and Large Eddy Simulation (LES) model may be used. The combined model reduces to the standard SST model in high mean shear regions (e.g., near viscous walls), where the grid is refined and has a large aspect ratio unsuitable for LES models. As the grid is traversed away from high mean shear regions, it typically becomes coarser and more isotropic, and the combined model smoothly transitions to an LES model. The intent is to improve predictions of complex flows in a real-world engineering environment, by allowing the use of LES methods with grids typical of those used with traditional Reynolds Averaged Navier Stokes models.

The Chien k-ϵ model is also available in WIND. Several options may be specified with this model to control the initialization procedure, enhance its stability, and improve its accuracy in adverse pressure gradients and at high Mach numbers.

Keywords: TURBULENCE

3.3.4 Transition Specification

Through the use of the TTSPEC keyword, point-by-point transition data may be specifed on viscous walls. The data represent the percentage of turbulent viscosity to be added to the laminar viscosity at each grid point. An auxiliary code, *tmptrn*, may be used to create the transition data, and write it into the common flow (*.cfl*) file.

Keywords: TTSPEC

3.4 Gas Models

A variety of gas models are available in WIND to complete the equation set. The fluid may be treated as a thermally and calorically perfect gas, a thermally perfect gas (frozen chemistry), equilibrium air, or a mixture undergoing a finite rate chemical reaction. Several different chemistry packages are available as files containing thermodynamic data, finite rate coefficients, and transport property data.

Keywords: CHEMISTRY

3.5 Other Models

3.5.1 Actuator Disks

To simulate fan or compressor discontinuities, WIND provides an actuator disk modeling capability, which acts as a modification to the zone coupling boundary condition. The model assumes an infinitesimally thin disk and must be applied at a coupled zonal interface. The actual discontinuity is specified in the input data file as a solid-body rotation or free vortex flow, the effects of which are applied when transferring flow information between the two specified zone boundaries.

Keywords: ACTUATOR

3.5.2 Screens

Flowfield screens may also be modeled in WIND as discontinuities across coupled zonal interfaces. In the WIND input data file, one must specify the zones and boundaries between which the screen in located, the solidity of the screen, and one of several methods for calculating the losses through the screen. The screen model is not intended for use with choked screens, where the screen is significantly limiting the mass flow rate. During the solution start-up phase, it may be necessary to specify a low solidity, then increase it to the desired value to avoid strong choking in transients.

Keywords: SCREEN

3.5.3 Vortex Generators

A model is available in WIND for the effects of an array of vane-type vortex generators in three-dimensional flow. Currently, the generators must be located at a coupled i-interface boundary between two zones. A discontinuous change in secondary velocity is applied across the interface to simulate the vortices produced by the generators.

The model is based on experimental data, and determines the strength of each vortex based on the generator chord length, height, and angle of incidence with the incoming flow, as well as the incoming flow core velocity and boundary layer thickness. Each vortex center is placed at the grid point closest to the location determined by the user-specified generator location on the boundary, and the generator height.

Keywords: VORTEX GENERATOR

3.6 Flowfield Initialization

By default, WIND initializes the computational flowfield by setting the flow properties at each grid point equal to those specified with the FREESTREAM keyword in the input data file. The same initial conditions are applied at all points in the computational domain, including solid walls, zone boundaries, and freestream boundaries. Several options in WIND lead to non-default initializations, including user-specified inflow conditions, boundary layer initialization, and reinitialization of portions of the flowfield on restart.

3.6.1 User-Specified Initialization

WIND's ARBITRARY INFLOW keyword was designed to specify flow conditions at arbitrary inflow boundaries.[11] However, it may also be used in a variety of ways to specify initial conditions in selected portions of the computational domain that are different from the freestream values. This capability may be useful, for example, when modeling a jet emanating from a solid wall. Zones downstream of the jet exit may have difficulty converging to the proper solution from conditions much different than the jet exit conditions.

To use the ARBITRARY INFLOW keyword as an initialization tool, specify the appropriate parameters and values for a zone as described below, and run WIND from scratch (i.e., without an existing flow (*.cfl*) file).[12]

There are three ways that the ARBITRARY INFLOW keyword may be used to modify the default initial conditions.

- If uniform inflow is specified for a particular zone using the UNIFORM parameter, the flow *throughout that zone* will be initialized to the conditions specified.
- The IJK_RANGE parameter may be used to set flow conditions over an arbitrary range of i, j, and k indices in any computational region. Up to 500 IJK_RANGE parameters are allowed. The default initial conditions will be used at points outside the specified range(s).
- The USERSPEC parameter may be used to specify a 1-D profile normal to the surface, translated through some buttline range, below the vehicle. Note that this option only applies to points in the $i = 1$ computational plane. The default initial conditions will be used at the remaining points.

Keywords: ARBITRARY INFLOW

[11] Recall that the type of boundary, such as **arbitrary inflow**, is specified using GMAN, and stored in the common grid (*.cgd*) file, not in the input data file.

[12] Note that since the ARBITRARY INFLOW keyword is also used to specify boundary conditions at arbitrary inflow boundaries, a conflict arises if (for some reason) the desired inflow properties are different from those being set as initial conditions. In this case, the ARBITRARY INFLOW keyword can still be used to set initial conditions by setting the number of cycles to be run to zero. Then, after changing the values specified with the ARBITRARY INFLOW keyword to the desired inflow values, simply restart using the initialized flowfield in the newly-created *.cfl* file.

3.6.2 Boundary Layer Initialization

To provide a better approximation to near-wall flowfields, WIND provides a couple of boundary layer initialization options that may help speed the convergence of viscous flows near solid walls.

First, by using multiple `IJK_RANGE` parameters with the `ARBITRARY INFLOW` keyword, and setting the range to include not only the inflow boundary but also locations downstream, one can set initial boundary layer profiles all along the viscous walls. Note that using this capability may add many lines to the input data (*.dat*) file. The `INCLUDE` keyword may be useful in keeping the main *.dat* file to a manageable size.

Another option is to use the `BL_INIT` keyword to specify a starting location and thickness for a laminar or turbulent boundary layer. However, this option may only be used on computational j- or k-boundaries, and only on one boundary in each zone.

Keywords: `ARBITRARY INFLOW`, `BL_INIT`

3.6.3 Reinitialization

In the event that portions of the computed flowfield become "polluted" with unrealistic flowfield data, due to numerical instabilities or other causes, you may wish to reinitialize portions of the flow. WIND's reinitialization option enables you to reset the flow conditions in specified zones. The reinitialization will be done as if WIND were being run from scratch. The conditions at all or specified grid points in the specified zones will be reinitialized to either freestream values, or to values specified using the `ARBITRARY INFLOW` keyword and/or the `BL_INIT` keyword, as described above.

Keywords: `REINITIALIZE`, `ARBITRARY INFLOW`, `BL_INIT`

4 Numerical Modeling

WIND utilizes a number of different numerical methods in the solution of a selected set of field equations. By default, the code solves the Euler and Navier-Stokes equations using Roe's flux-difference splitting scheme for the explicit (right-hand side) terms, and either a diagonalized or block matrix solver for the approximately-factored implicit (left-hand side) terms.

4.1 Iterations and Cycles

The usual method of obtaining a solution with WIND involves the initialization of the entire flowfield and the successive iteration of the flow solver, with appropriate explicit boundary conditions, to a steady-state solution. Most boundary conditions are updated after each iteration of the flow solver; however, because of its expense, zone coupling takes place less frequently. Zone coupling is the process whereby WIND exchanges flowfield information between zones to maintain solution continuity at zonal interfaces. Updates of the flow file with the most current data also take place less frequently.

To facilitate flow file (see Section 7.3) updates and propagation of flowfield information to neighboring zones, WIND defines a "cycle" as a specific number of iterations performed in each of the zones, with zone coupling to effect information transfer. By default, WIND performs five iterations in each zone before moving on to the next zone. You must specify the number of solution cycles to be computed, and you may optionally specify the number of iterations per cycle to be calculated in each zone (or all zones simultaneously).

For single-zone problems, you may wish to specify a large number of iterations per cycle, because there is no need for zone coupling operations; however, note that the flow file will be updated less frequently in this case. For multi-zone problems, a large number of iterations per cycle is not recommended, because it will greatly hinder the transfer of flowfield information between zones and, therefore, convergence.

4.1.1 "Turning Off" Zones

To "turn off" a zone in a multi-zone calculation, simply specify 0 or fewer iterations per cycle for that zone; values of -1 and -2 are used to indicate how zone coupling is to be handled for zones adjacent to the "dead" zone.

Keywords: `CYCLES`, `ITERATIONS`

4.2 Grid Considerations

WIND provides capabilities which allow you to make the most efficient use of the computational grid for a given configuration. Both of the options described in this section reduce the accuracy of the computations in one or more grid directions and should therefore not be used haphazardly.

4.2.1 Grid Sequencing

Grid sequencing is a capability whereby WIND "removes" grid points from selected portions of the domain, resulting in a "coarse grid" solution which is obtained in a fraction of the time it would have taken for a "full grid" solution. For example, if you select one level of sequencing in all three

directions in a particular zone, only every other grid point in each of the i-, j-, and k-directions will be used in the calculations, resulting in a decrease in CPU time of approximately 80–90%. (The speed-up is not a linear function of the number of sequencing levels.) Selecting two levels of sequencing would repeat the process, causing only every fourth grid point to be used. At the end of each run, the solution is interpolated back onto the entire grid to aid in restarting the solution and to provide a continuous field for post-processing.

Grid sequencing is very useful for converging gross flowfield properties before obtaining a final solution on the full grid. Note, however, that a sequenced solution may seem incorrect when post-processed using the entire computational grid, because of the interpolation process used at the end of each run. In addition, when restarting a full-grid solution from a sequenced solution, reducing the CFL number (time step) by 50% is highly recommended, in order to reduce the risk of solution instabilities which arise during the transition.

Keywords: `CFL#`, `SEQUENCE`

4.2.2 Thin-Shear-Layer Calculations

Another time-saving option available in WIND is the ability to solve the thin-shear-layer Navier-Stokes equations, instead of the full equations. By requesting that WIND remove the viscous terms from the equations in one or more computational directions, a CPU savings of up to 30% may be realized. This capability requires careful specification, because viscous terms must be retained along grid lines which are perpendicular to boundary layers, free shear layers, and other highly viscous flow features.

Because some solution instabilities have been linked to the use of the thin-shear-layer option, it is recommended that you disable this option near the completion (convergence) of your solution, creating a fully viscous flowfield model.

Keywords: `TSL`

4.3 Explicit Operator

The first-derivative (non-viscous) terms on the right-hand side of WIND's equation set are calculated by the code's explicit operator. You may choose from a wide variety of explicit operators, independently of the implicit (left-hand-side) operator. An explicit operator is selected by its type and accuracy. Types include a central difference operator, the upwind Coakley, Roe, Van Leer, and HLLE schemes, and modified versions of the Roe, Van Leer, and HLLE schemes for stretched grids. Depending on the type of scheme used, the accuracy may be specified as anywhere from first to fifth order. The default explicit operator is Roe's second-order upwind-biased flux-difference splitting algorithm, modified for stretched grids.

Keywords: `RHS`

4.3.1 Explicit Smoothing

Certain types of explicit operators — for example, the central difference operator — may require the addition of numerical smoothing in order to dampen instabilities which are a natural part of the scheme. Smoothing must also be added explicitly when utilizing WIND's convergence acceleration capability. Values for second- and fourth-order smoothing coefficients and a smoothing limiter coefficient may be specified in the input data file.

Keywords: ACCELERATE, SMOOTHING

4.3.2 Explicit Boundary Damping

In flowfields dominated by acoustic phenomena, unwanted wave reflections may sometimes occur at freestream computational boundaries. To eliminate these unphysical reflections, WIND's boundary damping capability may be employed to apply explicit smoothing near the freestream boundaries, with the effect of absorbing the reflected waves.

Keywords: BOUNDARY-DAMP

4.3.3 Total-Variation-Diminishing (TVD) Operator

Certain types of explicit operators are equipped with a Total-Variation-Diminishing (TVD) operator, which limits local maxima and minima in flux quantities to prevent non-physical instabilities from arising during the solution. The default Roe scheme, for example, requires interpolation and/or extrapolation of flux quantities, which may produce minima or maxima that are outside the range of the original data. TVD operators attempt to limit these peaks to acceptable values based on the data used in the interpolation and extrapolation. Three different TVD operators are available in WIND, and may be specified on a zonal basis. For two of the operators, the severity of the limiting may be controlled through an input parameter.

Because only single data points are used in the first-order schemes, TVD, which limits interpolations and extrapolations, has no effect on such cases. It will only have an effect for second- and higher-order calculations. When attempting to model precise phenomena, such as acoustic waves, TVD should *not* be used, because it may produce undesirable damping or smearing of flowfield features. However, in most cases, TVD provides significant protection against numerical instabilities with less than a 10% increase in CPU time.

Keywords: TVD, BOUNDARY TVD

4.4 Implicit Operator

The left-hand side of WIND's equation set is computed in the code's implicit operator, which may be set on a zonal basis. Options are available to select scalar or full block implicit operators, or to turn off the implicit operator completely for an entirely explicit calculation. With these options, a different implicit operator may be specified for each computational direction.

By default, WIND uses the full block operator in viscous directions, and the scalar (diagonalized) matrix solver in inviscid directions.

A variety of other implicit operators are also available, including point Jacobi, Gauss-Seidel, MacCormack's modified approximate factorization, and the ARC3D 3-factor diagonal scheme as used in the OVERFLOW code.

Keywords: IMPLICIT

4.4.1 Implicit Boundaries

In addition to the normal explicit boundary conditions, implicit boundary conditions may be used on "wall" boundaries. These improve stability when the CFL number is above about 1.3.

Keywords: IMPLICIT BOUNDARY

4.5 Time Step

4.5.1 CFL Number

The time step can be one of the most important parameters in a WIND solution. It may mean the difference between a successful run and a code bomb. In WIND, the time step is represented by the CFL (Courant-Friedrichs-Lewy) convergence criterion, which is a non-dimensional time step based on the actual time step, the local grid spacing, and the local characteristic velocity $(u + c)$.

For stability reasons, WIND's default CFL number is set to 1.3, which is sufficient for a wide variety of problems. For inviscid solutions, the CFL may be raised slightly, while for highly complex flows (due to complex geometry or gas models), it may need to be lowered significantly. Often, in the beginning of a solution, or during the transition between a sequenced and fine grid solution, the CFL number should be set to 0.5 while transients settle out of the flowfield.

Note: If you must consistently set the CFL number to 0.1 or lower to maintain stability in the solution, you should attempt to locate other possible causes of numerical instability, such as grid anomalies or grid/flow interactions. Such low CFL numbers translate into small time steps and unacceptable convergence rates.

When a CFL number above 1.3 is desired, the implicit boundary conditions should be used.

By default, WIND defines the CFL number based on the minimum time step associated with the eigenvalues of each coordinate direction individually. That is:

$$dt = \min\left(\frac{CFL_x}{\lambda_x}, \frac{CFL_y}{\lambda_y}, \frac{CFL_z}{\lambda_z}\right)$$

Using TEST 105, the user may define the time step more conventionally as:

$$dt = \frac{CFL}{\lambda_x + \lambda_y + \lambda_z}$$

which provides a somewhat smaller effective time step.

Options in WIND allow the CFL number or time step to be gradually increased as the calculation proceeds, or to use the time step calculation procedure from the OVERFLOW CFD code.

For most calculations, the time step is computed based on a constant CFL throughout the field, which means that, for non-uniform grids, the solution advances at different rates in different parts of the grid. Because the CFL number is directly proportional to the physical time step and inversely proportional to the local grid spacing, the physical time step must be small where the grid spacing is small, in order to maintain a constant CFL number. For example, in boundary layers and shear layers, where the grid is often closely packed for better resolution, the solution advances more slowly than in other parts of the flowfield. In fact, most of the iterations in a viscous solution are spent converging viscous regions. In practice, the major flow features (e.g., shocks and other major pressure gradients) will develop quickly, while detailed flow phenomena will require a larger number of iterations.

WIND therefore uses by default a "cross-flow" CFL number which multiplies each coordinate direction's eigenvalues by a factor before determining the time step associated with that spacing and flow conditions. This factor varies from 1.0 when the flow is along the coordinate direction, to a user-specified value when the flow is perpendicular to the coordinate direction. The far-field CFL

number is generally the value specified in the input data file with the CFL# keyword, but near the wall the code effectively uses a larger value. This should increase convergence of the boundary layer.

The cross-flow CFL factor is defined using the CROSSFLOW keyword. Values greater than one effectively increase the time step in the boundary layer (where the flow is parallel to the wall, but the minimum spacing is normal to the wall) by the specified factor. The default is 2.0 for three-dimensional flow, and 1.0 (i.e., no time step increase in the boundary layer) for two-dimensional flow and axisymmetric flow.

Keywords: CFL#, CROSSFLOW

4.5.2 Runge-Kutta Time Step

Several Runge-Kutta time step formulations are available for use in WIND. A traditional four-stage Jameson-type scheme or a three-stage, minimal-storage scheme may be selected in the input data file. You may also specify algorithm coefficients for each of the stages in the different schemes. Runge-Kutta schemes attempt to provide a more accurate approximation to the solution at the next time level by advancing through several stages, or sub-iterations, to get there. In WIND, each sub-iteration may be one pass through the explicit operator with or without a pass through the implicit operator. (The original Runge-Kutta scheme was formulated using only an explicit operator.) Depending on the coefficients used, Runge-Kutta time step algorithms are often higher order in time than the default Euler time stepping algorithm. This makes them ideally suited for time-accurate calculations (see below).

Keywords: STAGES

4.5.3 Global Newton Iteration

A Global Newton iteration technique is available in WIND, and has proven to provide time-accurate solutions with large CFL numbers for unsteady flows with large time scales (Tramel and Nichols, 1997). The Global Newton technique stabilizes the solution and improves time accuracy by placing the entire unsteady transport equations on the right-hand side of the matrix solver, and iterating within a time step over all the zones. Thus, the interface boundaries are brought up to the new time level, along with the interior flow field, resulting in an essentially implicit treatment of the boundaries. The Global Newton algorithm has also been shown to improve steady-state convergence rates (Nichols and Tramel, 1997).

Keywords: NEWTON

4.5.4 Time-Accurate Solutions

The objective of time-accurate CFD calculations is the modeling of unsteady flow phenomena, such as vortex pairing or bluff body wakes. In these calculations, the physical time step must be the same throughout the entire flowfield, regardless of the local grid distribution. Because the CFL number has a limit which approximates the upper limit of numerical stability, the time step used in the calculation must be obtained from the maximum CFL number and the smallest grid spacing in the field. Thus, the stability of the calculations in regions of dense grid packing determines the time step for the entire rest of the field. For single-zone time-accurate calculations, you may request that WIND calculate the minimum time step in the zone based on the specified CFL number. However, for multi-zone time-accurate calculations, you must calculate the time step yourself, based on the

desired CFL number and the minimum grid spacing, and specify the physical time step in the input data file.

Keywords: CFL#

4.6 Convergence Acceleration

The solution transients that occur in iterating from the freestream-initialized flowfield often cause the most severe stability problems in WIND. As the solution approaches convergence on a sequenced grid, most of these transients will have settled out of the field, with the change to fine grid being the only remaining "jolt" to the solution. Numerical stability may be forced through the addition of artificial numerical dissipation to the equations, but such dissipation may lead to non-physical results in the steady state solution.

WIND's convergence acceleration capability was developed to make use of artificial dissipation to eliminate stability problems, while raising the initial solution time step to get past the transients more quickly. In the input data file, you may specify the initial amount of artificial dissipation (smoothing) to be added to the equations, the starting and ending values of the CFL number, the iteration numbers between which the smoothing should be applied and CFL number altered, and the zone(s) in which you would like to enable this capability. Recommended values for the smoothing coefficients are given later in this manual, under the ACCELERATE keyword description (see p. 83).

At the beginning of the solution, using this capability, a relatively high CFL number will be accompanied by significant numerical smoothing. As the iteration count approaches the specified ending value, the CFL number will be gradually adjusted toward its final value as the amount of smoothing is steadily decreased.

Keywords: ACCELERATE, SMOOTHING, TEST 49 2

5 Boundary Conditions

The boundary conditions are perhaps the most important factor in influencing the accuracy of the flow computation. The manner in which the boundary conditions are imposed also influences the convergence properties of the solution. WIND uses a cell-vertex discretization, which results in solution points located on the boundaries of the zones which comprise the flow domain. During the computation, WIND computes the boundary values for the conservative variables, the species (if present), and turbulence variables (if present).

Proper specification of the flow boundary conditions is aided by a basic understanding of characteristic theory of the incoming and outgoing waves normal to the boundary. The boundary normal is considered positive when it points into the flow domain. The wave speeds (eigenvalues) have convective and acoustic components. A positive wave speed indicates a wave entering the flow domain, and so, a physical boundary condition must be specified and some auxiliary information must be supplied to impose the boundary condition. A negative eigenvalue indicates a wave leaving the flow domain, and so, a numerical boundary condition can be specified using flow data within the domain. The waves moving tangential to the boundary are neglected in the boundary condition treatment.

5.1 Explicit and Implicit Boundary Conditions

WIND imposes the boundary conditions explicitly after the interior solution has been computed for each zone after each iteration, by default. An exception is the zone interface boundaries, which are updated after each cycle. Another exception is the mass flow boundary condition, which is updated after a specified number of iterations (the default is five iterations) in order to reduce computational effort. Errors due to explicit boundary conditions are reduced through the use of the multi-stage or iterative time integration methods.

The use of implicit boundary conditions with an implicit solver is known to improve the stability of the method and lead to faster convergence at the expense of greater computational effort and complexity. Implicit boundary conditions are available in WIND in a limited manner, primarily for wall boundary conditions. The `IMPLICIT BOUNDARY` keyword is used to turn on implicit boundary conditions.

Keywords: `IMPLICIT BOUNDARY`

5.2 Boundary Condition Types

The Grid MANipulation (GMAN) program is used to associate boundary condition types with the solution points on the boundaries of the zones. These boundaries represent the geometry model, fluid boundaries, grid topological boundaries, and couplings between zones. The type of boundary conditions available in GMAN, in the order displayed by GMAN, include[13]:

- `undefined`
- `reflection`
- `freestream`
- `viscous wall`
- `arbitrary inflow`

[13] In this User's Guide, GMAN boundary condition types are indicated by lower-case words in a fixed-pitch font, `like this`; WIND keywords are displayed in an upper-case fixed-pitch font, `LIKE THIS`.

- outflow
- inviscid wall
- self-closing
- singular axis
- coupled
- bleed
- pinwheel axis
- frozen
- chimera

It is possible to group some of these boundary condition types in a logical manner for detailed discussions, which are presented below. Some boundary conditions require further information, which is provided through keywords in the input data file, beyond being flagged in GMAN.

5.3 Wall Boundary Conditions

The inviscid wall, viscous wall, and bleed boundary condition types are all wall boundary conditions which simulate interaction of the flow with a real or imaginary solid surface.

5.3.1 Inviscid Wall

The `inviscid wall` boundary condition imposes flow tangency at the zone boundary (wall surface) while maintaining the same total velocity as the point adjacent to the boundary. One numerical boundary condition is imposed by computing the pressure at the boundary through an interpolation of interior pressures. A zero-order extrapolation is robust; however, the pressures may not be smoothly varying at the boundary. A first-order extrapolation works well for flows without discontinuity, and for flows in which the pressure does not vary greatly normal to the boundary. The extrapolations across a discontinuity may result in nonphysical pressures. One can use the `viscous wall` boundary condition along with the `TURBULENCE INVISCID` keyword in the input data file. This is useful if one wants to start a computation with the boundary as inviscid and then later turn on the viscous boundary conditions. In WIND, it is also possible to specify through `TEST 138` that the normal pressure gradient at the wall be calculated rather than simply assuming it to be zero. Also, a first-order extrapolation which accounts for grid spacing can be used through the use of `TEST 141`.

5.3.2 Viscous Wall

The `viscous wall` boundary condition imposes a no-slip condition of the flow, a zero pressure gradient, and the appropriate heat transfer condition (adiabatic or constant temperature) at the zone boundary (wall surface). The no-slip condition can involve a non-zero velocity if the wall is moving. (See the `MOVING WALL` and `ROLL` keywords; however, for moving walls, the boundary condition type should be set to `bleed`.) To minimize transients at the start of a WIND calculation, the velocity at no-slip boundaries is actually reduced from its initial value to the no-slip condition over a number of iterations. The number of iterations may be specified using the `WALL SLIP` keyword.

The choice of the heat transfer condition is determined through the use of the `WALL TEMPERATURE` keyword in the input data file. The default is an adiabatic wall (zero temperature gradient). The

temperature for the constant temperature condition is specified through the `WALL TEMPERATURE` keyword. The `TTSPEC` keyword is available for specifying a point-by-point distribution of surface temperatures.

Wall function boundary conditions may be used at viscous walls, using the White-Christoph law of the wall, through the `WALL FUNCTION` keyword. This feature is currently available for single-species flows only.

In WIND, it is also possible to specify through `TEST 138` that the normal pressure gradient at the wall be calculated rather than simply assuming it to be zero. Also, a first-order extrapolation which accounts for grid spacing can be used through the use of `TEST 141`. Viscous flow is computed when the `TURBULENCE` keyword is used.

Keywords: `MOVING WALL`, `TTSPEC`, `TURBULENCE`, `WALL FUNCTION`, `WALL SLIP`, `WALL TEMPERATURE`

5.3.3 Bleed

The `bleed` boundary condition allows mass to flow through a porous, viscous wall. Bleed is mass flow out of the flow domain, while blowing is mass flow into the flow domain. Bleed and blowing systems are often an integral part of aeropropulsion configuration design, helping to control such flow phenomena as boundary layer growth and mixing. WIND's bleed / blowing boundary condition was designed to provide a means to model these systems with CFD. Specification of the `bleed` boundary condition in GMAN, which involves the identification of a particular bleed region number, triggers the calculation of the area of the bleed region specified, which is then stored in the grid file. WIND uses this area and the bleed or blowing conditions specified in the input data file to compute a normal velocity on the model surface. Bleed may be specified as a mass flow, or as a porous surface with a discharge coefficient and back pressure. Blowing may be specified through mass flow, plenum, or valve conditions.

Keywords: `BLEED`, `BLOW`

Moving Wall. The `moving wall` boundary condition enables a tangential velocity to be applied at no-slip walls in order to model rotating hubs or other components in the flow. The boundary solution points for the moving wall should be identified as `bleed` regions in GMAN. The translating or spinning motion of the wall is specified through the `MOVING WALL` keyword in the input data file. The `ROLL` keyword allows a rolling motion to be imposed on the grid.

Keywords: `MOVING WALL`, `ROLL`

5.4 Flow Interface Boundary Conditions

The `freestream`, `arbitrary inflow`, and `outflow` boundary condition types form a group involving the simulation of the interaction of the flow with other flow conditions at the domain boundaries.

5.4.1 Freestream

The `freestream` boundary condition is intended for use at freestream outer boundaries. This boundary condition uses one-dimensional characteristic theory to set boundary flowfield variables from freestream or flowfield conditions, based on the flow direction at the boundary.

At freestream boundaries with subsonic inflow, the HOLD keyword may be used to specify whether total conditions or characteristic values are to be held constant. Total pressure is held constant in both cases, and may result in initial Mach numbers being altered.

For some cases, this boundary condition may also be used at unconfined outflow boundaries, but the `outflow` boundary condition described in Section 5.4.3 is generally recommended instead. In particular, experience has shown that using the `freestream` condition at outflow boundaries with a shear layer exiting the computational domain may result in very slow convergence, and the solution may not be very accurate. There may also be the possibility of a mixed boundary, such as a supersonic outflow with a small subsonic region in the wall boundary layer. In this case, the pressure in the supersonic region is extrapolated to the boundary from upstream, but the pressure in the subsonic region is set from the freestream value. This may cause a small disturbance at the boundary, which can be corrected by specifying the boundary as an `outflow` boundary and imposing a constant pressure for the entire boundary.

Keywords: HOLD

5.4.2 Arbitrary Inflow

The `arbitrary inflow` boundary condition allows conditions to be specified on regions of zonal boundaries where flow is entering the zone. Such a capability may be required to describe a thermally stratified nozzle input flow, or a jet emanating from a wall. The inflow profile may be specified in a number of different ways: as uniform flow, as a point-by-point (xyz) profile, or as uniform flow over a range of grid indices. The ARBITRARY INFLOW keyword is used in the input data file to indicate desired flow properties.

Keywords: ARBITRARY INFLOW

5.4.3 Outflow

The `outflow` boundary condition may be used for internal flows at boundaries where subsonic flow is leaving the computational domain, such as at the exit plane of an inlet, diffuser, or auxiliary flow duct. It is also recommended for downstream outflow boundaries in external flow problems, especially if a shear layer is exiting the computational domain.

Characteristic theory indicates that only one physical condition is required to define the boundary condition. One may know one of the following at the outflow boundary: mass flow, exit pressure, or exit Mach number.

The mass flow (in lb_m/sec) may be specified at the outflow boundary (see the MASS FLOW keyword). The mass flow may be the actual or corrected value. One may alternatively specify the ratio of the desired mass flow to the mass flow through the inflow capture area specified in GMAN. During the solution, the mass flow boundary condition is applied every five iterations by default, which reduces computational costs. The integrated mass flow is compared with the desired value. If the `PRESSURE` option is used with the MASS FLOW keyword, a spatially-constant pressure is set at the outflow boundary, and modified as the solution proceeds until the desired mass flow is achieved. If the `DIRECT` option is used, the momentum, and thus the mass flow, is modified directly, and the pressure adjusts as the solution proceeds. For the `PRESSURE` option, WIND displays the ratio between the computational and desired mass flows and the modified pressure at each application of the boundary condition. If you experience difficulty in converging the mass flow, you should consider setting a constant back pressure at the duct exit.

A constant exit pressure may also be specified (see the DOWNSTREAM PRESSURE keyword) at the outflow boundary. This option results in a very reflective boundary condition, which may cause difficulties in convergence of the solution, especially for internal flows. One alternative is to allow the exit pressure to vary spatially according to the distribution of the solution points adjacent to the boundary. This option is selected through the VARIABLE option of the DOWNSTREAM PRESSURE keyword. The UNSTEADY option of the DOWNSTREAM PRESSURE keyword may be used to specify either a sinusoidal or user-defined pressure oscillation at an outflow boundary.

The mass-averaged Mach number may be specified at the outflow boundary using the DOWNSTREAM MACH keyword. This boundary condition is identical to the Chung-Cole compressor face boundary condition discussed below. It simulates the uniform Mach number characteristics that have been observed experimentally at compressor faces. This boundary condition also corresponds to a fairly uniform mass flux through the outflow boundary.

The compressor face models of Chung and Cole (1996) and of Mayer and Paynter (1994) are also available at outflow boundaries, through the COMPRESSOR FACE keyword. Both models are based on the observation that turbine engine conditions set the corrected mass flow, and that this corresponds directly to the average Mach number at the compressor face. These boundary conditions have been implemented mainly for the analysis of unsteady flow; however, they have also been shown to be robust for the establishment of steady-state, supercritical inlet flows.

The computational grid at the outflow boundary should be such that it is modeled with a single computational plane (constant i, j, or k) in a single zone. If two zones merge together near the exit, one should create a small exit zone to accommodate the outflow boundary condition.

Keywords: COMPRESSOR FACE, DOWNSTREAM MACH, DOWNSTREAM PRESSURE, MASS FLOW

5.5 Grid Topology Boundary Conditions

The reflection, self-closing, singular axis, and pinwheel axis types of boundary conditions form a group involving the simulation of the flow at topological surfaces in the grid.

5.5.1 Reflection

The reflection boundary condition simulates a plane of symmetry, and is the same as a solid, slip wall boundary condition. Therefore, within WIND, the inviscid wall boundary condition is actually applied. The reflection boundary condition type does provide a descriptive label for the boundary solution points, which may be of use to several auxiliary CFD codes which generate or use reflected grids.

5.5.2 Self-Closing

The self-closing boundary condition can be used at boundaries for which grid lines connect end-to-end (e.g., i_{max} connecting to i_1 as in an O-grid) in a point-match manner. The boundary condition simply averages the flow variables from both sides of the boundary and assigns the average to the two boundaries. This condition was formerly used for polar-type grids in duct flowfields. Note: For best results, one should consider using the coupled boundary condition instead of self-closing.

5.5.3 Singular Axis

The `singular axis` boundary condition is imposed at locations where an entire or part of a zonal boundary has collapsed to a line (*not a point*). Thus, along the other boundary direction, the grid points are coincident. The singular grid point is evaluated by taking the distance-weighted average of the solution of the adjacent grid points encircling the axis. This boundary condition is not to be used when the singularity line collapses to a point. For partially singular boundaries, the average is computed only over the singular portion of the boundary. Excessive use of this boundary condition is not recommended since flow conservation is not preserved. One should attempt to use non-singular grids whenever possible. `TEST 118` allows the choice of which variables are averaged. `TEST 150` can be used to explicitly set certain velocity components to zero when the singular axis occurs on symmetry planes. `TEST 199` excludes the last grid point on the singular axis from being averaged.

5.5.4 Pinwheel

The `pinwheel axis` boundary condition is used when there exists multiple singularities at various locations on boundaries. This boundary condition does not zero any velocity components — it simply takes the average. `TEST 118` allows the choice of which variables are averaged. `TEST 165` allows input of the integer for the order of magnitude of the tolerance for singularity (the default is 10^{-8}).

5.6 Zonal Interface Boundary Conditions

The `coupled` and `chimera` boundary condition types involve the interactions between zones of the domain. Periodic boundaries are treated as a type of `coupled` boundary condition.

5.6.1 Coupled

The `coupled` boundary condition is imposed at regions where zones connect. Zone coupling is the name of the process by which WIND transfers flowfield information from one zone to another across conterminous computational grid boundaries. This process uses geometric interpolation factors stored in the grid file, which have been previously computed in the GMAN program. The zone interfaces occur at the boundaries of the zones, and so are imposed as boundary conditions at the end of each cycle of the computation.

The default zone coupling algorithm is based on Roe's flux-difference splitting scheme. The `COUPLING` keyword allows one to change a zone coupling algorithm based on one-dimensional characteristic theory.

Zone boundaries should be treated the same as you would treat interior grid planes; there should not be any large changes in grid stretching or orthogonality at the boundary. In addition, zone boundaries should not be placed in regions of strong flowfield gradients, especially horizontally along jet shear layers or aligned with normal shocks. In other words, use your best engineering judgement in placing zone boundaries in your solution.

Keywords: `COUPLING`

Periodic Boundaries. Periodic boundaries are treated as normal coupled boundaries in WIND, with the connection data stored in the common grid (*.cgd*) file when setting boundary conditions with GMAN. No keyword is needed in the input data (*.dat*) file.

GMAN's CONNECT MODE option is used to specify how to move one boundary surface to overlay it on the corresponding periodic boundary surface. There are two connection modes — translation and rotation. For translation, the user defines the Δx, Δy, and Δz movement for the surface. For rotation the user sets the center of rotation and the angles. GMAN then moves the current boundary surface using the specified method, and attempts to couple the two surfaces together as if they were coexistent boundaries. If successful, it saves the data as normal coupling data for WIND. Thus the boundaries look connected even though they are physically separated. The periodic boundary surfaces do not have to be in the same zone, nor do they have to be point matched. They only have to line up physically once the movement has been performed.

Specifically, to set a periodic boundary condition in GMAN:

- In graphics mode, from the main menu select BOUNDARY COND.
- Pick the zone and boundary for one of the two periodic boundaries.
- If the boundary condition at the boundary is not UNDEFINED (you may select IDENTIFY POINTS to check), change it to UNDEFINED by doing:
  - Select MODIFY BNDY.
  - Select CHANGE ALL.
  - Select UNDEFINED.
  - From the main menu, select BOUNDARY COND.
  - Select YES-UPDATE FILE.
- Select MODIFY BNDY.
- Select COUPLE.
- Select SEL OTHER BND.
- Pick the zone and boundary for the other periodic boundary. (Note that no default zone is pre-selected here, even for single-zone grids, so you must select both the zone and boundary.)
- Select SET COUP MODE.
- Under CONNECT MODE, click on the "** NONE **".
- Respond to the prompts at the bottom of the screen to set the connection mode, and to specify the translation or rotation data.
- Select CONNECT (not COUPLE) to actually connect the two boundaries.

5.6.2 Chimera

The chimera boundary condition indicates that the zone boundary is in an overlap region.

5.7 Miscellaneous Boundary Conditions

The undefined and frozen boundary condition types are included in this section, mostly because they don't fit well in any other section, and are fairly self-explanatory.

5.7.1 Undefined

The default boundary condition type in GMAN is `undefined`. If not set to another boundary condition type, the boundary solution point is evaluated as an average of local solution points. In practice, one should specify the actual boundary condition type for all boundary solution points and avoid having undefined boundary solution points.

If all the points on a boundary surface have an undefined boundary condition, an error message is printed and the solution will abort. If only some points have an undefined boundary condition, a warning message is printed and the solution will continue.

5.7.2 Frozen

The `frozen` boundary condition simply signals a boundary solution point to retain its value as read in from the initial solution file.

6 Convergence Monitoring

Monitoring and properly assessing convergence levels during a WIND run are critical in obtaining meaningful, useful results. WIND allows you to track convergence by following residuals and/or integrated forces, moments, and mass flow. For engineering applications, the recommended convergence monitoring method is the tracking of integrated quantities of interest. For example, if you are modeling a wing/body geometry to determine drag, you should monitor integrated drag and set some reasonable bounds on drag oscillations as your convergence criterion.

6.1 Residuals

In steady-state (non-time-accurate) solutions, the residuals are the amounts by which the solution vector changes in a single iteration. Ideally, in approaching a steady state solution, the residuals should approach zero. However, in practice, complex geometric and flowfield features may limit the reduction in the residuals to about two orders of magnitude. WIND prints solution residuals to the list output file, in order to get a general idea of solution convergence. For each zone, WIND prints the zone number, cycle number, location of the maximum residual (i, j, and k indices), equation number for which the maximum residual occurred, the value of the maximum residual, and the L2-norm of all the residuals for all the equations over all the points in that zone. By default, the residuals are printed each iteration. The output interval may be changed, however, using the `CYCLES` and `ITERATIONS` keywords. Residuals may be plotted by first using the auxiliary program *resplt* to generate a GENPLOT file, then using that file as input to the `plot` command in the CFPOST post-processing package.

The L2-norm of the residuals will give you an idea of the overall convergence of the solution. The location of the maximum residual may give you insight into problems with a particular solution. After solution "bombs" or when convergence is unacceptably slow, the location of the maximum residual is the first place you should look for potential problems with the solution.

If the maximum solution residual decreases by four orders of magnitude from its maximum value in a particular zone, WIND will display a "converged" message in the list output file and stop iterating in that zone. Note that, if the maximum residual is high due to large initial solution transients, WIND may decide that a zone is converged when it may require significantly more iterations. In the input data file, you may specify the number of orders of magnitude required before WIND's convergence check is triggered, and you may also set the absolute value of the maximum or L2-norm residual to be checked for convergence.

Keywords: `CONVERGE`, `CYCLES`, `ITERATIONS`

6.2 Integrated Flowfield Quantities

As noted above, the optimal convergence checking criterion for a solution is an integrated quantity of interest for a particular study. For example, for an afterbody drag study to calculate pressure drag, the integrated drag is obviously the best quantity to track in monitoring convergence. This may be accomplished using WIND's flowfield integration capability. In the input data file, you may specify a number of computational surfaces for which WIND will integrate forces, moments, and/or mass flows. If you specify reference length, area, and moment center, WIND will output the forces and moments as coefficients; you may even specify output of lift and drag coefficients in place of x and y force coefficients.

You may specify zones and computational grid indices of subsets to be integrated, and you may

request that different quantities be integrated on each subset. You may also control the frequency of output and whether the code outputs subset totals and/or zonal and grand totals. The auxiliary program *resplt* will produce GENPLOT plot files of output integration quantities versus solution cycle number, which may then be used as input to the `plot` command in the CFPOST post-processing package.

Keywords: `LOADS`

6.3 History Tracking of Flow Data

In time-accurate calculations, where a constant time step is being specified and is the same throughout the flow field, you may wish to track a particular flow variable as the solution advances in time. For example, you may be interested in monitoring the static pressure on a backward-facing step as vortices roll off the back of the step. You may request history tracking in the input data file. Specification must include zone numbers and ranges of grid points to track, and the frequency of the sampling. When you request history tracking, WIND creates a time history file containing the sampled data. The auxiliary program *thplt* must be run to extract data from the history file into GENPLOT plot files, which may then be used as input to the `plot` command in the CFPOST post-processing package.

Keywords: `HISTORY`

7 Files

A number of files are used by WIND in the course of a solution. The script file you run to submit WIND jobs will assign all the necessary files to their appropriate Fortran unit numbers, so you should not need to do any of that yourself. Each of the support files is described briefly below.

7.1 Input Data File (*.dat*)

Fortran unit number 5

The *input data file* is the primary control file for WIND and must be created for each case you want to run with the code. Input data and code options are entered through the use of descriptive keywords in this file. You should observe the following formatting rules in creating the input data file:

1. The first three lines of the file are reserved for geometry, flow condition, and arbitrary titles, respectively. Each of these titles may be up to 64 characters long. None of these first three lines may start with the (case-insensitive) word "Include".
2. Comment lines, beginning with a '/' character, may be placed anywhere in the file after the first three lines. Data file readability may be improved dramatically through the liberal use of comments — for example, separating logical sections of the data file: numerical algorithm, force integrations, test options, etc.
3. Block data, such as that specified in arbitrary inflow or chemistry cases, must be contiguous — only keywords corresponding to the block may reside between the beginning and ending block indicators (e.g., `CHEMISTRY` and `ENDCHEMISTRY`).
4. Keywords may be entered in upper or lower case.
5. Abbreviations for keywords may be used, as long as they are unique. If they are not, you may not get the results you expect. For example, it's not a good idea to use single-letter abbreviations for keywords.

The following is an example of a simple input data file:

```
Geometric Title
Flowfield Condition Title
Optional Title
Freestream static 0.9 14.7 530. 4. 0.
Cycles 15
Inviscid
End
```

7.2 Grid File (*.cgd*)

Fortran unit number 11

The computational grid used by WIND for a particular case is stored in the *grid file*. In this file are stored the (x, y, z) coordinates of all computational grid points, zone coupling interpolation factors, and grid reference and scaling data.

This file was originally referred to as the *common* grid file, so named because the file was formatted according to Boeing's Common File Format (CFF). Starting with version 4.0 however, WIND also supports grid files in CGNS (CFD General Notation System) format.[14] Both common files and CGNS files are binary, and portable to virtually every hardware platform, except Cray, with no need for explicit data conversions.

Grid files for WIND may be created by several mesh-generation codes in either common file or CGNS format. If necessary, the *cfcnvt* utility may be used to convert a variety of other formats, including PLOT3D xyz format, to common file format.

Zone coupling, reference, and scaling data are added to common grid files using the GMAN program. Common grid files are also used in the CFPOST post-processing package. Neither GMAN nor CFPOST currently support CGNS files, however.

Keywords: CGNSBASE

7.3 Flow File (*.cfl*)

Fortran unit number 20

The *flow file* contains the computed flow field. For Navier-Stokes and Euler solutions, the file contains density, momentum, and energy data, and, for viscous solutions, turbulence data. The flow file also records the current solution cycle number to allow the file to be used for solution restarts.

Like the grid file, the flow file was originally referred to as the *common* flow file, but may now be written in either common file or CGNS format. Several graphical post-processing programs are able to read both common files and CGNS files.

The CFPOST post-processing package may be used with common flow files to produce other files for post-processing and/or to create flowfield plots directly. CFPOST does not currently support CGNS files, however.

Keywords: CGNSBASE

7.4 Global Newton File (*.cfk*)

Fortran unit number 16

The *Global Newton file* is a copy of the flow file used for Global Newton time marching. It holds the current sub-iteration of the time step. It is only created when Global Newton time marching is active, and is a scratch file. If the code aborts, the solution restarts from the conditions in the flow file, at the beginning of the time level, so the Global Newton file does not need to be saved.

7.5 Boundary Data File (*.tda*)

Fortran unit number 14

WIND's *boundary data file* is used during solution restarts, and results in smoother restarts, especially with higher-order boundary coupling. The file acts as a buffer for the transfer of zone coupling information and a holding bin for data needed throughout a WIND run but not stored in the grid or solution file.

[14] Detailed information on the CGNS standard may be found at the CGNS web site, at http://cgns.sourceforge.net/.

7.6 Time History File (*.cth*)

Fortran unit number 19

The *time history file* is a common file which stores data resulting from the use of WIND's history tracking capability. The file contains a lookup table corresponding to the range of computational indices tracked during the run, time stamp data, and flowfield data for each time stamp. Upon completion of a time history run, the auxiliary program *thplt* may be used to view the contents of the time history file.

Keywords: `HISTORY`

7.7 List Output File (*.lis*)

Fortran unit number 6

The purpose of WIND's *list output file* is to echo the input from the input data file, track convergence and integration results, record CPU/job statistics, and log code error messages. The auxiliary program *resplt* may be used to extract convergence and integration data from the list output file and create a GENPLOT-style data file, which may be plotted with the `plot` command in the CFPOST post-processing package. For parallel-processing runs, this file also contains messages relating to the PVM system and to the allocation and operation of slave processors. Convergence data is written to this file for each of the equation sets solved during the run, including the Euler, Navier-Stokes, and non-algebraic turbulence model sets.

7.8 WIND Stop File (*NDSTOP*)

Fortran unit number 1

The *stop file* is used to stop WIND execution cleanly in the middle of a CFD solution. Although a solution may be stopped simply by killing the WIND process from the system queue, doing so will not ensure a clean update of all zonal flowfield data to the flowfield file. The stop file — named *NDSTOP* — provides a means to cleanly shut down a running solution, completing the current cycle or zone, performing zone coupling, updating the flowfield file, and removing all symbolic links to Fortran file unit numbers.

To stop the code, you must place one of two words in the stop file in the directory in which the WIND job is running. The word `STOP` in the stop file signals WIND to complete the current cycle (at the end of the last zone) and exit. In single-processing mode, you may also use the word `STOPZONE` in the stop file to stop WIND after completing the zone currently in memory.[15] You may then restart the code in your next run, starting at the next zone or back at zone one. Regardless of the existence of this file, WIND will stop if the requested number of cycles has been computed, or if your solution has converged.

During the clean-up procedure at the end of the job, the *NDSTOP* file is automatically removed.

Note that the directory in which the WIND job is running, where the *NDSTOP* file must be, is not necessarily the one you were in when the *wind* command was issued. If you don't use the *-runinplace* option to the *wind* script (see Section 8.1), the job will be run in a remote directory. The root name of the remote run directory may be specified using the *-runroot* option, or in response to a command line prompt. The default for the "root name" of the remote directory is */tmp*. The

[15] In multi-processing mode, `STOPZONE` does the same thing as `STOP`.

full name of the remote directory will be *rootname/userid/basename.scr*, where *rootname* is the root name described above, *userid* is your userid, and *basename* is the base name of your *.dat* file.

On a Unix system, you might submit a simple 'at' job for a later time as follows:

```
at 0530 monday
echo STOP > NDSTOP
^D
```

At 5:30 AM on the next Monday, the system would create the *NDSTOP* file with the word '`STOP`' in it. Note that this will not stop the run exactly at 5:30 AM; WIND must still complete the current cycle, which may take an hour or more for large cases.

Keywords: RESTART

7.9 Temperature and Transition Specification Files

Fortran unit number 45

WIND can read temperature and transition files, specifying the temperature or transition to turbulence on any boundary surface, that were created with the older *tmptrn* utility. This option is included for backward compatibility with WIND 2.0, and is only needed for an initial (i.e., non-restart) run. The *wind* script copies the files to the run directory, and WIND opens the files directly (unit 45) using the file names specified with the TTSPEC keyword.

New applications should use the latest *tmptrn* utility to write the temperature/transition data into the common flow (*.cfl*) file directly.

Keywords: TTSPEC

7.10 Chemistry Files (*.chm*)

Fortran unit number 26

The generalized chemistry files contain all the information WIND requires to compute a general chemistry mixture. A chemistry file has a header line, followed by three sections containing thermodynamic coefficients, finite rate coefficients, and transport properties.

7.10.1 Header

The header line must contain a single parameter, *ispec*, specifying the type of reaction rate and the format for the finite rate data. The format for the header line is[16]:

ispec ISPEC

where

[16]The parameter *ispec* is read using a list-directed read statement. The `ISPEC` label following the value is thus optional, but present in the standard chemistry files supplied with the WIND code. The labels following the parameters *ns*, *nreq*, *ndeq*, and *tfrmin*, described in Section 7.10.2 and Section 7.10.3, are also optional.

ispec	Reaction Rate and Format
1, 4, 100, 110	Forward and backward rates from equilibrium constant polynomial, Format 1
3, 130	Forward and backward rates, Format 2
120	Forward rate only, Format 3

7.10.2 Thermodynamic Coefficients

This section of the chemistry data file contains the information necessary to compute the specific heat c_p for each species. The value of c_p is defined by a series of polynomials, each valid within a defined temperature range, of the form:

$$c_p = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4$$

where T is in K, and c_p is in J/kg-K.

The format for this section is:

```
THERMODYNAMIC COEFFICIENTS
Title, line 1
Title, line 2
ns      NS
Information defining species 1
...
Information defining species ns
```

where *ns* is the number of species.

For each species, the information defining the species and for computing c_p, etc., is read as follows:

```
      read (26,100) name,ncurves,ds1,hm1,ds2,hm2,unk1,unk2,con,molwt
      do 10 ic = 1,ncurves
         read (26,110) t1,t2,(a(i,ic),i=1,5),shf(ic),F(ic)
   10 continue
  100 format (a8,7x,i5,4x,2(a2,f3.0),16x,f10.1,5x,f10.3)
  110 format (5e15.5)
```

where

Variable	Definition
`name`	Name of species
`ncurves`	Number of temperature segments in c_p vs. T curve
`ds1`	Name of first constituent atomic species in `name`
`hm1`	Number of atoms per molecule of `ds1`
`ds2`	Name of second constituent atomic species
`hm2`	Number of atoms per molecule of `ds2`
`unk1`	Unknown (0.0)
`unk2`	Unknown (0.0)
`con`	Coefficient defining number of degrees of freedom, defined as (translational degrees of freedom + rotational degrees of freedom)/2.
`molwt`	Molecular weight
`t1,t2`	Beginning and end of temperature range for curve segment `ic`
`a(1-5,ic)`	Polynomial coefficients a_1–a_5 for curve segment `ic`
`shf(ic)`	Heat of formation of species, in J/kg
`F(ic)`	Gibb's free energy (for future use in gas constant)

7.10.3 Finite Rate Coefficients

This section of the file contains the reaction rate information. There are three possible formats for this section, depending on the value of *ispec*.

Format 1; *ispec* = 1, 4, 100–119. This option may be used for any chemically reacting flow. Information is specified defining the forward reaction rate k_f and the equilibrium constant K. The rates themselves are computed using the equations

$$k_f = CT^S e^{-D/(\kappa T)} \tag{1}$$

$$k_b = k_f / K \tag{2}$$

$$K = \exp\left(a_1 + a_2 Z + a_3 Z^2 + a_4 Z^3 + a_5 Z^4\right) \tag{3}$$

$$Z = 10,000/T \tag{4}$$

where T is the temperature in K, the ratio D/κ is in K, and κ is the Boltzmann constant. The reaction rate coefficient C has units $(\text{cm}^3/\text{g-mol})(\text{sec-K}^S)^{-1}$.

The various parameters and polynomial coefficients are read from the chemistry data file, in the following form:

```
FINITE RATE COEFFICIENTS
Title, line 1
Title, line 2
nreq ndeq       NREQ, NDEQ
tfrmin          TFRMIN
Information defining dissociation reaction 1
...
Information defining dissociation reaction ndeq
Information defining exchange reaction 1
...
```

Information defining exchange reaction nreq - ndeq

where *nreq* is the total number of reactions (dissociation + exchange + ionization), *ndeq* is the number of dissociation reactions (i.e., have a third body), and *tfrmin* is the temperature in K below which no reactions occur.

For each reaction, the information defining the reaction and for computing the reaction rates is read as follows:

```
      read (26,100) r1,r2,p1,p2,S,D,(a(i),i=1,5),nthd
      do 10 i = 1,nthd
         read (26,110) spec,C
   10 continue
  100 format (4(a5,3x),2e12.4/5e12.4,i4)
  110 format (16x,a8,e12.4)
```

where

Variable	Definition
`r1`	Name of reactant 1
`r2`	Name of reactant 2. (Leave blank for dissociation reactions.)
`p1`	Name of product 1
`p2`	Name of product 2
`S`	Temperature exponent S in equation (1)
`D`	D/κ in equation (1)
`a(1-5)`	Polynomial coefficients a_1–a_5 in equation (3)
`nthd`	Number of third bodies, implies `nthd` reactions (one for each third body). Always set `nthd` = 0 for exchange and ionization reactions.
`spec`	Name of third body reactant (not needed if `nthd` = 0 or 1)
`C`	Third body coefficient for species `spec` as third body, or average if `nthd` = 0 or 1

With this option, the data is read in subroutine `frtin1`, and the rates are computed in `rates1` (for *ispec* = 1), `rates4` (*ispec* = 4), `rates` (*ispec* = 100), or `ratesa` (*ispec* = 110). [At some point, the special-case options *ispec* = 1 and 4 may be eliminated in favor of one of the more general *ispec* = 100–119 options.]

Format 2; *ispec* = 3, 130–139. This option is similar to the previous one, except that the forward and backward reaction rates are computed separately, using the equations

$$k_f = C_f T^{S_f} e^{-D_f/(\kappa T)} \tag{5}$$

$$k_b = C_b T^{S_b} e^{-D_b/(\kappa T)} \tag{6}$$

where T is the temperature in K, the ratios D_f/κ and D_b/κ are in K, and κ is the Boltzmann constant. The reaction rate coefficients for the j'th reaction, $(C_f)_j$ and $(C_b)_j$, have units

$$(C_f)_j\text{:}\quad \left(\frac{\text{cm}^3}{\text{g-mol}}\right)^{O_j-1} \frac{1}{\text{sec-K}^{S_f}}, \qquad (C_b)_j\text{:}\quad \left(\frac{\text{cm}^3}{\text{g-mol}}\right)^{O_j-1+\nu_j} \frac{1}{\text{sec-K}^{S_b}}$$

where O_j is the order of the reaction (i.e., the total number of moles of reactants), and ν_j is the number of moles of products minus the number of moles of reactants.

The various parameters are read from the chemistry data file, in the following form:

```
FINITE RATE COEFFICIENTS
Title, line 1
Title, line 2
nreq ndeq       NREQ, NDEQ
tfrmin          TFRMIN
Information defining dissociation reaction 1
...
Information defining dissociation reaction ndeq
Information defining exchange reaction 1
...
Information defining exchange reaction nreq - ndeq
```

where *nreq* is the total number of reactions (dissociation + exchange + ionization), *ndeq* is the number of dissociation reactions (i.e., have a third body), and *tfrmin* is the temperature in K below which no reactions occur.

For each reaction, the information defining the reaction and for computing the reaction rates is read as follows:

```
      read (26,100) r1,nr1,r2,nr2,p1,np1,p2,np2,Sf,Df,Cf
      read (26,110) Sb,Db,Cb
  100 format (4(a5,f3.1),3e12.4)
  110 format (32x,3e12.4)
```

where

Variable	Definition
`r1,nr1`	Name and number of reactant 1
`r1,nr2`	Name and number of reactant 2. (Leave blank for dissociation reactions.)
`p1,np1`	Name and number of product 1
`p2,np2`	Name and number of product 2
`Sf,Sb`	Temperature exponents S_f and S_b in equations (5) and (6)
`Df,Db`	D_f/κ and D_b/κ in equations (5) and (6)
`Cf,Cb`	Third body coefficients C_f and C_b in equations (5) and (6)

With this option, the data is read in subroutine `frtin3`, and the rates are computed in `rates3` (for *ispec* = 3), or `ratesb` (*ispec* = 130). [At some point, the special-case option *ispec* = 3 may be eliminated in favor of one of the more general *ispec* = 130–139 options.]

Format 3; *ispec* = 120–129. This option is only for global 1- or 2-reaction chemistry, and only for forward reactions. It is a quick method for simulating detonation problems, for example, in which reactions proceed only forward to completion. The forward reaction rates are computed using the equation

$$k = CT^S e^{-D/(\kappa T)} \tag{7}$$

where T is the temperature in K, the ratio D/κ is in K, and κ is the Boltzmann constant. The reaction rate coefficient for the j'th reaction, C_j, has units $(\text{cm}^3/\text{g-mol})^{O_j-1}(\text{sec-K}^S)^{-1}$, where O_j is the order of the reaction (i.e., the total number of moles of reactants). There are no backward reactions.

The various parameters and polynomial coefficients are read from the chemistry data file, in the following form:

```
FINITE RATE COEFFICIENTS
Title, line 1
Title, line 2
nreq ndeq       NREQ, NDEQ
tfrmin          TFRMIN
Information defining reaction 1
...
Information defining reaction nreq
```

where *nreq* is the total number of reactions (dissociation + exchange + ionization), *ndeq* is the number of dissociation reactions (i.e., have a third body), and *tfrmin* is the temperature in K below which no reactions occur.

For each reaction, the information defining the reaction and for computing the reaction rates is read as follows:

```
      read (26,100) r1,nr1,r2,nr2,p1,np1,p2,np2,S,D,(a(i),i=1,5),nm
      read (26,110) Cb1,Cb2,C
  100 format (4(a5,f3.1),2e12.4/5e12.4,i4)
  110 format (2f8.3,8x,e12.4)
```

where

Variable	Definition
`r1,nr1`	Name and number of reactant 1
`r2,nr2`	Name and number of reactant 2
`p1,np1`	Name and number of product 1
`p2,np2`	Name and number of product 2
`S`	Temperature exponent S in equation (7)
`D`	D/κ in equation (7)
`a(1-5)`	Place holder for use with future *ispec* options
`nm`	Place holder for use with future *ispec* options
`Cb1,Cb2`	Exponents on concentrations of reactants 1 and 2
`C`	Third body coefficient C in equation (7)

With this option, the data is read in subroutine `frtin2`, and the rates are computed in `ratesf`.

7.10.4 Transport Coefficients

This section of the chemistry data file contains the coefficients used to compute the laminar viscosity and thermal conductivity. The formulation is based on Wilke's law which uses Sutherland's law individually for each species. Thus for each species, there are equations of the form:

$$\frac{\mu}{\mu_0} = \left(\frac{T}{T_{0\mu}}\right)^{\frac{3}{2}} \frac{T_{0\mu} + S_\mu}{T + S_\mu} \tag{8}$$

$$\frac{\kappa}{\kappa_0} = \left(\frac{T}{T_{0\kappa}}\right)^{\frac{3}{2}} \frac{T_{0\kappa} + S_\kappa}{T + S_\kappa} \tag{9}$$

The various parameters and polynomial coefficients are read from the chemistry data file, in the following form:

```
TRANSPORT COEFFICIENTS
```

Title, line 1
Title, line 2
Information for species 1
...
Information for species ns

For each species, the information is read as follows:

```
      read (26,100) name,ncurves,t1,t2,mu0,t0mu,smu
      read (26,110) k0,t0k,sk
      do 10 ic = 2,ncurves
         read (26,120) t1,t2,mu0,t0mu,smu
         read (26,110) k0,t0k,sk
   10 continue
  100 format (a8,2x,i5,2f10.3,1x,3e12.4)
  110 format (36x,3e12.4)
  120 format (15x,2f10.3,1x,3e12.4)
```

where

Variable	Definition
`name`	Name of species
`ncurves`	Number of temperature segments in μ and κ vs. T curve
`t1,t2`	Beginning and end of temperature range for curve segment `ic`
`mu0`	Reference viscosity
`t0mu`	Reference temperature in equation (8), K
`smu`	Reference temperature offset in equation (8), K
`k0`	Reference conductivity
`t0k`	Reference temperature in equation (9), K
`sk`	Reference temperature offset in equation (9), K

Keywords: CHEMISTRY

7.11 Reserved Files

Unit numbers 15, 21, and 55 are reserved for proprietary features, and should not be used by WIND developers.

8 Scripts

WIND is run using a Unix script which attaches appropriate files and either starts WIND interactively or submits the job to a queueing system. In addition, this script checks for special files and may be used to re-invoke WIND after performing intermediate operations.

8.1 *wind* — Run WIND Code

The usual procedure for running the WIND code is via a command-line script called *wind*. With this approach, the specific executable to be run (e.g., production version, alpha version, etc.), the names of the required input and output files, etc., may be defaulted, specified on the command line, or entered as responses to prompts. The various options to the script are described below in the form of a Unix-style man page for *wind*.

If the *Tcl/Tk* interpreter is available, the WIND code may be launched via an interactive GUI by using **-g** as the first option to the *wind* script, i.e., by typing `wind -g`. In this case, the executable to be run, the file names, etc., are specified via the GUI. Any options to the *wind* script following the **-g** are ignored. If the **-g** option is specified, but the *Tcl/Tk* interpreter is not available, the usual command-line interface will be used.

The *wind* Man Page

NAME

wind — a computational platform for solving various sets of equations governing fluid dynamics

SYNOPSIS

wind [**-batch**] [**-begtime** *time*] [**-binroot** *directory*] [**-cfdrootrem** *directory*] [**-dat** *datfile*] [**-debug**] [**-debugger** *debugger*] [**-endtime** *time*] [**-flow** *cflfile*] [**-g**] [**-grid** *cgdfile*] [**-grpcharge** *number*] [**-help**] [**-list** *lisfile*] [**-memory** *number*] [**-mp**] [**-mpmode** *mode*] [**-ncpu** *number*] [**-nice_val** *number*] [**-nobg**] [**-(no)parallel**] [**-not_nice**] [**-nzones** *number*] [**-program** *windname*] [**-queue_name** *name*] [**-remoterun**] [**-runinplace**] [**-runque** *queue*] [**-runroot** *directory*] [**-tmpdir** *directory*] [**-usessh**] [**-walltime** *time*]

DESCRIPTION

wind is a Bourne shell script used to run the WIND code, a computational platform for numerically solving various sets of equations governing physical phenomena. Currently, the code supports the solution of the Euler and Navier-Stokes equations of fluid mechanics, along with supporting equation sets governing turbulent and chemically reacting flows. See the *WIND User's Guide* for detailed information on the operation and use of the WIND code.

Internally, the *wind* script invokes other scripts, and creates on the fly a *job file* (with the suffix *.job*), which is the actual script submitted to the queue specified by the user.

OPTIONS

-batch Run in "batch" mode, with all necessary input specified via command-line options. The user must specify the WIND executable to be run using the **-program** option; all the input/output file names using the **-dat**, **-list**, **-grid**, and **-flow** options; the system run queue using the **-runque** option. The user must also specify either the **-runinplace** option, or the directory to run in using the **-runroot** option. In addition, if a multi-processing control (*.mpc*) file exists, either the **-parallel** or **-noparallel** option must be specified. And, if an *.mpc* file

exists and MPI message passing is being used, the number of zones must be specified using the **-nzones** option. If any of these items are missing, an error message will be displayed and execution will stop.

-begtime *time* — The beginning time for a standard Unix *at/batch* job (i.e., when using **-runque** AT_QUE).

-binroot *directory* — Root location of the WIND executable on the system on which it will be running.

-cfdrootrem *directory* — The "CFDROOT" directory on the system on which the WIND executable will be running. The default is the same as the "CFDROOT" directory on the local machine.

-dat *datfile* — The WIND input data (*.dat*) file, entered without the extension (e.g., *wing*, not *wing.dat*).

-debug — Run WIND in debug mode, using the debug package specified by the **-debugger** option. The job will automatically be run interactively, and the **-runinplace** option will be set.

-debugger *debugger* — Set the debug package to be used when running in debug mode. The default depends on the computer system being used.

-endtime *time* — Ending time for a Unix *at/batch* job, or time to run in seconds for a job running under the basic QSUB, PBS, or LSF queueing system. For the PBS and LSF queueing systems this is the total time for the entire job, minus some length of time for termination processing (i.e., updating the flow (*.cfl*) file, terminating worker processes, etc.) Also see the **-walltime** option.

-flow *cflfile* — The WIND flow (*.cfl*) output file, entered without the extension (e.g., *wing*, not *wing.cfl*). If the file exists, the solution will restart using the conditions in the file as initial conditions.

-g — When specified as the first option, use the *Tcl/Tk* interactive GUI to specify input parameters, ignoring any other options. If **-g** is not the first option specified, it is ignored.

-grid *cgdfile* — The WIND grid (*.cgd*) input file, entered without the extension (e.g., *wing*, not *wing.cgd*).

-grpcharge *number* — Group charge number for the job, used for NAS systems.

-help — Display the list of command-line options, and quit, ignoring any other options.

-list *lisfile* — The WIND list output (*.lis*) file, entered without the extension (e.g., *wing*, not *wing.lis*). Set the parameter *lisfile* to "screen" to display the list output file at the terminal.

-memory *number* — Amount of run-time memory to request for a job running under the PBS (in megabytes) or LSF (in kilobytes) batch system.

-mp — Run using a multi-processor machine (i.e., a single machine with multiple processors, like the CRAY or SGI Origin-2000), using either PVM or MPI message passing.

-mpmode *mode*
Message passing mode: either PVM or MPI. MPI may only be used on multi-processor systems. Note that the pre-compiled executables available through IVMS were built without MPI message passing. To use **-mpmode MPI**, MPI must be pre-installed on your system (unlike PVM, MPI is not distributed with WIND), and you'll need an executable that includes links to the MPI library. See the section "Installing the Build Distribution" in the *WIND Installation Guide* for instructions on creating the executable. The default message passing mode is PVM.

-ncpu *number*
Number of CPUs to request for a job running under the PBS or LSF batch system.

-nice_val *number*
The "nice" value (a number from 1 to 20) to use with the Unix *nice* command. Larger values cause the job to run at a lower CPU scheduling priority to lessen the impact on other jobs on the system. For interactive jobs (**-runque** REAL), the value is automatically set to 2. For jobs run in parallel mode, the master task, and worker tasks being run on the same system as the master, automatically run at top priority (i.e., without using *nice*).

-nobg
Jobs in the interactive queue normally run in the foreground if the output is being displayed on the screen, and in the background if the output is being sent to the *.lis* file. This option specifies that the interactive job is to be run in the foreground, even if the output is being sent to the *.lis* file.

-(no)parallel
Run in parallel mode (or serial mode if **-noparallel** is specified). For interactive and *at/batch* jobs, parallel mode requires a multiprocessing control (*.mpc*) file. If this option is not specified, and an *.mpc* file exists, the user will be asked if parallel mode should be used.

-not_nice
Run without using the Unix *nice* command.

-nzones *number*
Number of zones, used in MPI message passing mode on multi-processor systems.

-program *windname*
The name of the WIND executable to be used.

-queue_name *name*
The name of the specific queue in which the job is to run. The form of the *name* parameter depends on the queueing system being used, as specified using the **-runque** option.

-remoterun
Bypass some of the built-in error checking for valid path names preventing error messages when running on a remote system. The "CFDROOT" directory on the remote machine should be specified using the **-cfdrootrem** option, and the directory to run in should be specified using the **-runroot** option.

-runinplace
Run in place, i.e., in the directory in which the *wind* script was invoked. The default is to run in a remote directory.

-runque *queue*
Queueing system in which the WIND executable is to run. The parameter *queue* must be "REAL" for a real-time interactive job; "AT_QUE" to run using a standard Unix *at* or *batch* queue; "QSUB_QUE" to run

in a basic QSUB queue; "QSUB_PBS_QUE" to run in a QSUB queue using the PBS batch system; or "BSUB_QUE" to run in a BSUB queue using the LSF batch system.

-runroot *directory* — Root directory under which the WIND executable is to be run. The full name will be *directory/userid/basename.scr*, where *userid* is your userid and *basename* is the base name of your *.dat* file. The default *directory*, if neither **-runroot** or **-runinplace** is specified, is */tmp*.

-tmpdir *directory* — The directory for storing temporary files during a WIND run. The default is the directory specified using the **-runroot** option when running in a remote system, and the current directory when using the **-runinplace** option.

-usessh — Use *ssh* remote shell/copy commands. Note that the pre-compiled executables available through IVMS were built assuming *rsh* (*remsh* for HP and Cray systems) will be used as the "remote shell" command for parallel operation. If your system requires the use of *ssh*, you'll need an executable that includes links to an "*ssh* version" of PVM. See the section "Installing the Build Distribution" in the *WIND Installation Guide* for instructions on creating the executable.

-walltime *time* — Total wall clock time in seconds for a job running under the PBS or LSF queueing system. This includes the length of time for termination processing (i.e., updating the flow (*.cfl*) file, terminating worker processes, etc.) Also see the **-endtime** option.

NOTES

Please send documentation suggestions/corrections to towne@grc.nasa.gov. Questions concerning the WIND code itself or the NPARC Alliance should be sent to nparc-support@info.arnold.af.mil

8.2 *wind_post* — Perform Post-Processing

At the end of a non-debug job that completes successfully, if a script named *wind_post* exists in the current directory, and if the job is not being run interactively, the *wind_post* script is invoked, as follows.

`wind_post` *datname lisname cgdname cflname iter run_command*

The six arguments are defined as:

Argument	Definition
datname	The base name (i.e., without the three-letter extension) of the input data (*.dat*) and job files
lisname	The base name of the list output (*.lis*) file
cgdname	The base name of the grid (*.cgd*) file. (If CGNS files are being used, note that this is the base name of the file itself, not the name of the `CGNSBase_t` node in the file.)
cflname	The base name of the flow (*.cfl*) file
iter	The "job iteration" number. This is a count of the number of jobs, not time step iterations. It is initialized to zero, and incremented by one before each invocation of *wind_post*. Note that the value of *iter* is not saved between runs. The next invocation of the top-level *wind* script will re-initialize the value to zero.
run_command	The command used to submit the job

The *wind_post* script itself must be supplied by the user. Users with some experience in writing shell scripts may use this feature to automatically post-process results at the end of a job, or to save and/or process interim results from a lengthy calculation and submit a new job.[17]

An example *wind_post* Bourne shell script called *wind_post.sh* is supplied with the WIND application distribution in the directory *wind/bin*. (There's also an example C shell version called *wind_post.csh*.) The example script resubmits the job with a different input file for each job iteration. It looks for an input data file named *datname.dat.nextiter*, where *datname* is the base name of the *.dat* file and *nextiter* is the *next* job iteration number (i.e., one more that the value of the input argument *iter*). The new input data file, if it exists, is *copied over* the original input data file and the job file is resubmitted.

A typical method for using this capability would be to create files named, for example, *test.dat.1*, *test.dat.2*, *test.dat.3*, etc. Then, copy *test.dat.1* to *test.dat* and submit the job as usual. By keeping the initial input file in *test.dat.1* and copying it to *test.dat*, the original initial input file is preserved.

After the initial job completes, *wind_post* will be called with *iter* = 1. The *next* job iteration number is thus 2. The file *test.dat.2* will be copied into *test.dat*, and the job will be resubmitted, restarting WIND where it left off. If the file *test.dat.nextiter* doesn't exist, all the input data files have been used. The job file is then removed, and the run ends.

Note the distinction between a *run* and a *job*, and the value of the job iteration number. By *run*, we mean here a single invocation by the user of the *wind* script. Within that run, multiple *jobs* may be submitted using the *wind_post* feature, each with a different, automatically-incremented job iteration number. As noted above, the job iteration number is not saved between runs. The next time the *wind* script is invoked, starting a new run, the job iteration number will be re-initialized to zero. If it's necessary to save the value between runs, the user-supplied *wind_post* script should save it to a file, to be read during the next run.

The example *wind_post* script does not do any post-processing, but that can easily be added. For example, CFPOST could be run, with input redirected from a command file, to create plot files or reports. Or, an interim flow (*.cfl*) file could be saved by copying it into a file with the *iter* variable as part of the file name.

[17] Note that a similar capability may be invoked using the `SPAWN` keyword in WIND's input data file.

8.3 *windver* — Get WIND Version Number

The alpha version of WIND, and to a lesser extent the beta version, change quite frequently. Even the released production version is sometimes modified, as a result of bug fixes. Changes are summarized in the "Source History" file, accessed through the Internet Version Management System (IVMS).

The *windver* script may be used to get the complete version number of a WIND executable, and (for WIND 3.0 and higher) its supporting libraries. For example,

```
% windver
Versions in ./SGIMP6.5/R12000/bin

                Select the desired version

  0: Exit wind
  1: alpha version
  2: Version 2.0
  3: Version 3.0
  4: Version 4.0
  5: Version 5.0

Enter number or name of executable.......[5]: 5
 WIND   - Version 5.193 (Last changed 2002/11/05 17:53:28)
 LIBCFD - Version 2.142 (Last changed 2002/10/14 17:27:00)
 LIBADF - Version 2.0.14.1 (Last changed 2002/07/08 17:21:40)
 PVM    - Version 3.4.3
```

windver lists the various WIND executables available, and prompts the user to select one of them. It then prints the complete version number, plus the date it was created, for the WIND code, the CFD_wind library (common file processing routines, system-dependent routines, and utility routines), the ADF library (ADF routines from the CGNS project), and the PVM library (PVM routines used for parallel processing). In the above example, WIND Version 5.0 is actually Version 5.193, created on Nov 5, 2002.

By default, *windver* looks for the WIND executables in the directory specified by the environment variable `WIND_BINROOT`. If there is no environment variable of that name, *windver* will look in `CFDROOT`. The user may also specify the root location of the WIND executable using the option *-binroot*. I.e., invoking *windver* as

```
windver -binroot /usr/local/wind/wind
```

will tell *windver* that the executables are located in the appropriate directories (i.e., in subdirectories corresponding to the `SYSTEM` and `SYSTEM_CPU` environment variables) below */usr/local/wind/wind*.

8.4 *windrun* — Quick WIND run

The *windrun* script may be used to quickly set up and start a WIND run. The run will be in interactive mode, with the output sent to a *.lis* file. The various options to the script are described below in the form of a Unix-style man page.

NAME

windrun — A quick way to start a WIND run

SYNOPSIS

windrun *casename* [**-dev | -cfd**] [*windname*] [**-serial | -parallel**] [**-new | -restart**] [**-help**]

DESCRIPTION

windrun is a Bourne shell script used to quickly set up and run WIND. Internally *windrun* invokes the *wind* script with the options *-runinplace*, *-nobg*, and *-runque REAL*, and other options determined by the *windrun* input options.

OPTIONS

casename	The prefix for the WIND input/output files. All files must have the same prefix (e.g., *case1.cgd*, *case1.cfl*, etc.)
-dev	Look under the directory defined by the *WIND_DEV* environment variable for the WIND executable. This is the default.
-cfd	Look under the directory defined by the *CFDROOT* environment variable for the WIND executable.
windname	The name (without the *.exe* extension) of the WIND executable to be used (e.g., *windalpha*, not *windalpha.exe*).
-serial	Run in serial mode. This is the default.
-parallel	Run in parallel mode, if an *.mpc* file is present.
-new	Run a new case from scratch. This is the default. Any existing *.cfl*, *.tda*, *.cth*, *.lis*, and/or *.job* files with the current *casename* in the current directory will be erased before starting the WIND run.
-restart	Start from the existing solution, if one exists.
-help	Display a summary on using *windrun*.

8.5 *windmp* — Run on Multi-Processor

The *windmp* script may be used as a shortcut when running WIND on a multi-processor machine (i.e., a single machine with multiple processors, like the CRAY or SGI Origin-2000). It simply invokes the *wind* script with the options *-mp*, *-runinplace*, and *-not_nice*.

8.6 *runtest, runtestsuite* — Run WIND Test Case(s)

runtest is a utility for running WIND test cases. *runtestsuite* is a wrapper for *runtest* that may be used to automate the running of a series of test cases. The various options to both scripts are described below in the form of Unix-style man pages.

The *runtest* Man Page

NAME

runtest — Run WIND test cases

SYNOPSIS

runtest *basedir casename* [**-dev | -cfd**] [*windname*] [**-serial | -parallel**] [**-baseline**] [**-help**]

DESCRIPTION

runtest is a Bourne shell script that may be used to run WIND test cases. The base name of the case to be run is given by the specified *casename*. *runtest* will look for *casename.dat* and *casename.cgd* input files in the current directory. In addition, *runtest* will recursively descend into any sub-directories and look for input files named *subdir.dat* and *subdir.cgd*, where *subdir* is the sub-directory name.

For example, you could set up a test case named *case1*, with input files *case1.dat* and *case1.cgd*. In the directory containing these files, you could have sub-directories named (say) *run1*, *run2*, etc., containing variations (e.g., different input options) on the original case. *runtest* will *cd* into the sub-directory *run1*, and look for input files named *run1.dat* and *run1.cgd*. Similarly for the sub-directory *run2*, etc. All such cases that are found will be run, in the directory containing the input files.

If a file named *casename.ic.cfl* (or *subdir.ic.cfl* if in a sub-directory, as described above) exists in the run directory, it will be copied as the file *casename.cfl* (or *subdir.cfl*) and used as initial conditions. Otherwise, a new case will be run from scratch. In either situation, any existing output files will be overwritten.

If multiple *.dat* files are found named *casename.1.dat*, *casename.2.dat*, etc. (or *subdir.1.dat*, etc.), multiple runs will be made using the successive *.dat* files to restart the solution.

If the specified case has been run previously, any convergence measures stored in the *.lis* file (i.e., residuals, integrated flow properties, etc.) will be compared with the values from the existing baseline results using *diff*, and the output will be written to the file *basedir/test.log*. The *-baseline* option may be used to replace an existing baseline solution with the current one, bypassing the comparison.

OPTIONS

basedir	The directory where the log file of the tests is to reside.
casename	The prefix for the WIND input/output files. All files must have the same prefix (e.g., *case1.cgd*, *case1.cfl*, etc.)
-dev	Look under the directory defined by the *WIND_DEV* environment variable for the WIND executable. This is the default.
-cfd	Look under the directory defined by the *CFDROOT* environment variable for the WIND executable.
windname	The name (without the *.exe* extension) of the WIND executable to be used (e.g., *windalpha*, not *windalpha.exe*). If no name is specified, the default WIND version will be used.
-serial	Run in serial mode. This is the default.
-parallel	Run in parallel mode, if an *.mpc* file is present.
-baseline	Replace the saved baseline solution, used for comparison with results from future runs, with the current solution.
-help	Display a summary on using *runtest*.

The *runtestsuite* Man Page

NAME

runtestsuite — Run a series of WIND test cases

SYNOPSIS

runtestsuite [**-dev | -cfd**] [*windname*] [**-serial | -parallel**] [**-baseline**] [**-help**]

DESCRIPTION

runtestsuite is a wrapper for the script *runtest* that may be used to automate the running of a series of test cases. It assumes that the cases to be run are stored in subdirectories of the current directory, with the subdirectory names the same as the case names (i.e., the subdirectory names are used for the *casename* argument when invoking the *runtest* script).

OPTIONS

-dev Look under the directory defined by the *WIND_DEV* environment variable for the WIND executable. This is the default.

-cfd Look under the directory defined by the *CFDROOT* environment variable for the WIND executable.

windname The name (without the *.exe* extension) of the WIND executable to be used (e.g., *windalpha*, not *windalpha.exe*). If no name is specified, the default WIND version will be used.

-serial Run in serial mode. This is the default.

-parallel Run in parallel mode, if an *.mpc* file is present.

-baseline Replace the saved baseline solutions for each case, used for comparison with results from future runs, with the current solutions.

-help Display a summary on using *runtestsuite*.

9 Parallel Processing

The time required to compute a solution can be reduced by using the distributed parallel processing capability of WIND. WIND can simultaneously use multiple systems connected via a network as though they were a single computer. The systems are typically workstation class machines and need not be all of the same vendor.

When operating in distributed parallel processing mode, the system on which the job originates is called the *master* system. All other systems are called *worker* systems. WIND distributes complete zones to participating worker systems. Each zone is solved in parallel with other zones on other systems. The systems exchange boundary information during each cycle in order to propagate information throughout the computational domain. There may be fewer systems than zones to be computed, in which case, a system will be assigned another zone when it finishes its current assignment. The user specifies the names of the participating worker systems via the *multi-processing control file*, described in Section 9.1.

If a worker system fails due to either a system or network failure during the course of a run, the job will restart from the last checkpoint (see the `checkpoint` directive in Section 9.1) without the failed system. The automatic restart ability will be invoked as many times as necessary during a job until no more systems are available.

9.1 Multi-Processing Control File

The multi-processing control file specifies the hosts that will be available as well as some miscellaneous options. The presence of a valid multi-processing control file is all that is required to utilize the distributed parallel processing capability of WIND. If the main WIND input data file name is *input.dat*, the name of the multi-processing control file must be *input.mpc*. When this file is present, the *wind* script will ask the user if they really want to use multi-processing mode.

Comments may be included in the file with the normal WIND comment indicator '`/`', or additionally '`#`'. Blank lines are ignored. Trailing comments are not allowed. The formats of the directives follow.

`host {localhost | `*hostname*`} [nproc `*n*`]`

`host` directives specify the names of the worker systems (given by the *hostname* parameter) that will be used to process zones. In general, there should be one `host` directive for each worker system. If a particular system appears more than once, each occurrence is treated as a unique system and will process assigned zones simultaneously. This is not advisable unless the system has multiple processors and sufficient memory.

The optional parameter `nproc` n may be used to specify the number of processes to allow to run in parallel on the specified host. It is equivalent to repeating the `host` directive n times.

If no `host` entries appear in the multi-processing control file, the originating system will automatically be selected as the only host. When used on a system with sufficient memory and the `assignment mode dedicated` directive, the normal I/O associated with a single processor solution will be eliminated (except for checkpoints).

The special parameter `localhost` is used when running on a single multi-processing system and the system name is unknown at the time of job submittal, such as for batch systems (like NQE) that

can spawn to multiple systems or clusters of servers. Using `localhost` is preferred over not putting in any `host` directives because it assures that the scripts set up WIND consistently.

`host` entries should appear in the file in decreasing order of computational power. WIND assigns the most computationally intensive zones to the highest entries in the list.

The system that originates the job is not automatically included in the host list. If it is desired to also assign solution tasks to the originating system, it should have a host entry like any other system. For estimating purposes, the master process typically consumes less than one percent of the CPU time on the master host.

`i/o {direct | indirect}`

This directive specifies the type of access that worker systems have to files on the master. The default is `indirect`, which means that workers do not have access to the files on the master, and that file I/O must therefore be done using message passing to/from the master process.

On multi-processor systems, however, `i/o direct` may be used to indicate that the worker processes may access the files directly, bypassing communication through the master process. This significantly reduces communication overhead and increases performance by as much as 10-40%.

There are a couple of things to be aware of when using the `i/o direct` option. First, it should only be used when running on a single multi-processor system, not with workstation clusters. And second, the number of open files per process must be set large enough.

`checkpoint {[every] {`*time* `minutes | `*count* `cycles} | none}`

This directive specifies how often the worker systems transfer their flow field information to the flow file on the master system. In the event of a failure, the solution is automatically restarted from the last checkpoint. Specifying too small a number can result in very high network overhead and low throughput. A large number improves performance but can cause wasted time if a lot of network failures occur. If `checkpoint none` is specified, the flow field information is updated only at the end of the job. The default value is:

```
checkpoint every 60 minutes
```

`assignment mode {dedicated | shared | transient}`

`assignment mode` controls how tasks are assigned to target systems. There may be multiple appearances of this directive. They affect subsequent `host` entries up to the next `assignment mode` directive. A description of each mode follows:

`dedicated` Each task (zone) gets a unique Unix process on the target system. If a system must process more than one zone, each will have a separate process, but only one will be allowed to run at a time unless multiple `host` entries are present for the system. This is the default mode and should not be changed unless there is insufficient swap space for the processes assigned to the host.

`shared` Unless a system must process more than one zone, this mode is the same as `dedicated`. If more than one zone must be processed, only one Unix process is allocated and individual zones are swapped to and from local disk on the target system. This mode should be used only if the target system does not have sufficient swap space to contain the zones it needs to process.

`transient` When a task completes, it writes all of its flow information back to the master processor and terminates. A new Unix process is then started for the next cycle. This mode is for future use and should not be specified.

`route [indirect | direct]`

Controls how data is routed between the master and worker tasks.

`indirect` All messages go from the task on the local machine to the local PVM daemon, over the network to the remote PVM daemon, which forwards it to the remote task.

`direct` All messages go directly from task to task, bypassing the PVM daemons. This mechanism should be more efficient, but cannot be used when the number of zones is large.

`#LOADLIMIT` *limit*

The WIND/PVM initialization script will automatically eliminate workers that are deemed "too busy." A system is defined to be "too busy" when its 15 minute load factor, as reported by the Unix "uptime" command (the last number on the line) is greater than a certain limit (0.60 by default). The purpose of this check is to eliminate hosts that are occupied by hung processes.[18]

The load factor for each worker will be displayed in the list output file at the top with the other messages that occur during the preparation of the workers. The load factor will be displayed as a percent (0.60 corresponds to 60%). Note that the load factors in excess of 100% are possible. A message will also be displayed if the load factor exceeds the allowed threshold.

Occasionally, there is a problem with the "uptime" command and it reports a high load factor when there is no load on the system. To avoid this problem, the `#LOADLIMIT` directive[19] may be used to override the default value of 0.60. The parameter *limit* specifies the load limit for all hosts up to the next `#LOADLIMIT` directive. A `#LOADLIMIT` directive with no parameters restores the default load limit. This command should only be used when you know that including an overloaded host will not affect your job.

The following example illustrates the use of the `#LOADLIMIT` directive in the multi-processing control file.

```
# Next statement considers hosts ws1463 and ws1464 loaded
#    only if their load factor exceeds 100%
#loadlimit 100
```

[18] Note that the load factor is checked only at initialization time. If the system becomes busy during a run . . . well . . . too bad.

[19] Note that this directive is an exception to the use of `#` as a comment indicator.

```
host ws1463
host ws1464
# The next statement restores the default load limit
#loadlimit
host ws1465
# Use a really high limit for ws1466 - disables the limit check
#loadlimit 9999
host ws1456
```

Another way to modify the default load limit is to set the PVM_LOAD_LIMIT environment variable *before* you submit your job. For example, using the Bourne shell:

```
$ PVM_LOAD_LIMIT=75
$ export PVM_LOAD_LIMIT
$ wind
```

9.2 Multi-Processing Script Considerations

Unix Security Considerations

The network communication software uses remote shell commands to start the worker tasks. This implies the user must have a valid account on the worker system. In addition, the user name on the master system and all worker systems must be the same.[20]

To allow remote shell commands to access a worker system, the host name of the master system must be in the file *.rhosts* in the user's home directory on the worker system, or in the system file *hosts.equiv* on the worker system. [If *ssh* is being used, the corresponding file names are *.shosts* and *shosts.equiv*. These files are used in exactly the same way as *.rhosts* and *hosts.equiv* and on a non-*ssh* system.] Note that this is required even if the master and worker are the same system.

The *.rhosts* file is a text file containing a list of system names, and the userids on each of those systems, that are allowed to access the current host via a remote shell command. The file should have its permissions set to `rw-----`, so issue the following command after creating the file:

```
chmod 600 .rhosts
```

Once the *.rhosts* file has been created, it may be tested by issuing the following command from the system where the job will be submitted.[21]

`rsh` *worker-name* `ls -la`

Things are functioning properly if the directory listing appears. In many situations the users home directory is located on a file server, so only one *.rhosts* file needs to be maintained.

More information about *.rhosts* files may be found by entering `man rhosts` on most Unix systems.

[20] By "remote shell commands" we mean *rsh* (*remsh* on HP and Cray systems) and *rcp*; or, if the *ssh* (secure shell) software is used, *ssh* and *scp*. Note that if *ssh* is to be used, that must be specified during the creation of the PVM and WIND executables. See the *WIND Installation Guide* for more information.

[21] Use the command *remsh* or *ssh* instead of *rsh*, if that's appropriate for your system.

Ensuring PVM Jobs Stop at a Specific Time

To avoid the possibility of a PVM job continuing outside an allotted off-shift window, a series of scripts can be executed by the Unix cron process. In the WIND distribution, these scripts are in the directory *wind/cfd/bin/pvmkill*. Four files are located there:

1. *cronkill* — This file tells the continuous running job scheduler when to terminate processes. The first two digits on each line are the minute, the third digit is the hour, and following the *'s are the days when each of the commands will be executed (Monday = 1). The first command is the 'nicest' way to kill the job, with the following two successively harsher. Note that this file must be edited so that output goes to your directory and the paths for the scripts are correct.
2. *pvmclean* — A script which terminates jobs in a relatively nice fashion.
3. *naskill* — A script which terminates jobs in a bit harsher fashion.
4. *naspvmkill* — A script which terminates jobs in the meanest fashion.

To invoke these processes, copy the above scripts to *each* master you're using, edit *cronkill* appropriately, and insert these processes into the crontab on *each* master by entering

```
crontab cronkill
```

To check if this worked, enter

```
crontab -l
```

which will give a list of all your cron entries.

9.3 Multiple Parallel Jobs

When running in parallel mode on a cluster of workstations, the master system and all worker systems being used by a given user cannot be used by any other parallel job from the same user as long as the first job is active. A different user, however, can have a parallel job running simultaneously on the same systems, assuming that the memory, disk space, etc., are sufficient to support multiple jobs.

There are no restrictions on the number of parallel jobs for a given user on a multi-processor system (i.e., using the *-mp* option to the *wind* script; see Section 8.1), again assuming that the computer resources are available to support multiple jobs.

9.4 Hints

Because synchronization takes place at the end of each cycle, total throughput is established by the processor that takes the longest to complete its assigned work. The optimum situation is to have all zones of equal size and have one processor for each zone. This gives maximum throughput and processor utilization, but is generally not achievable. If all zones cannot be close to the same size, a mixture of sizes is preferable. The case to avoid is a configuration with one zone of comparable size to the sum of the remaining zones. In this case, one can achieve at most a factor-of-two performance improvement regardless of the number of processors used. In general, if n is the number of points

in the largest zone and N is the total number of points, the maximum possible speed up is N/n (assuming identical processors and similar algorithm specification).

Given a number of processors P with relative speeds p_i (larger p implies faster), and a number of zones N of sizes n_j, the assignment of work is done as follows:

1. Assign the largest zone j to processor 1 and compute $T_1 = n_j/p_1$.
2. Repeat step 1 for the remaining $P - 1$ processors, assigning the largest remaining zone j to processor i and compute $T_i = n_j/p_i$.
3. If any zones remain to be assigned, locate processor i such that T_i is a minimum. Assign the largest remaining zone j to processor i, computing $T_i = T_i + n_j/p_i$.
4. Repeat step 3 for remaining unassigned zones.

Consider adding processors if T for any processor is significantly larger than the others, and that processor has more than one zone assigned.

The output file from a run will indicate what zones are assigned to what processor, and will have a report containing the utilization of each processor.

10 Keyword Reference

10.1 Text Conventions

In this manual, keywords are indicated by upper-case words in a fixed-pitch font, `LIKE THIS`. In the actual input file, however, they may be entered as upper or lower case. They may also be abbreviated. WIND checks for keywords in quasi-alphabetical order and does *not* check for uniqueness. Therefore, specifying "`A I`" for arbitrary inflow is not a good idea. In cases where multiple options are available, the default (if one exists) is underlined.

User-specified input parameters are indicated by italics, *like this*. These user-specified parameters are usually numeric values, but may also be other keywords.

Most keywords consist of a single line, containing the keyword and its user-specified input parameters. Other keywords, like `CHEMISTRY`, indicate the start of a keyword block containing several lines, bracketed by starting and ending keywords. Within a keyword block, only individual keywords relevant to that block may be used. Keyword blocks are indicated by the word "block" in parentheses. Some keywords have several options, making them long and complex. These are sometimes split into two (or more) lines for display purposes, with a "\" at the end of the line being continued, but must nevertheless be on a single line in the input data file.

The various keywords and keyword blocks are listed alphabetically, with each one starting a new page. The overall syntax for each keyword or keyword block is shown in a box at the top of the page, with the details following.

In addition, the following documentation conventions are also used:

`|`	The "or" symbol; used to separate multiple choices
`[]`	Delimiters surrounding optional entry(s)
`{}`	Delimiters surrounding multiple entries when exactly one of them is required

10.2 Zone Selection

Many of WIND's capabilities may be specified on a zone-by-zone basis. Keywords used to enable these capabilities may include a zone selector at the end of the keyword command in the input data file. The keywords for which this type of specification is valid include a *zone_selector* format specifier in their description. The *zone_selector* must be of the following form:

`[ZONE]` *zone-list*

where *zone-list* is of the form:

range1 `[,`*range2*`[,...`*rangen*`]]`

A *range* is of the form:

zonenum	Selects zone *zonenum*
begzone`:`*endzone*	Selects all zones from *begzone* to *endzone*
begzone`:`	Selects all zones from *begzone* to `MAXZONE`, the total number of zones in the grid file

`:`*endzone*	Selects all zones from 1 to *endzone*
`ALL`	Selects all zones

For some keywords that turn on capabilities that may be selected by zone, the *zone_selector* is optional (indicated by the syntax [*zone_selector*]). In this case, omitting the zone specification defaults to using the capability in all zones. You should therefore change the default selection before changing individual zones, as changing the default will reset any zones which have been individually set earlier in the input file.

Example

To turn off (i.e., perform no iterations) in zones 5, 6, and 7, but still pass the information in these zones to the adjacent zones, and to use 10 iterations per cycle in all other zones, use the following sequence of keywords:

```
ITERATIONS PER CYCLE 10
ITERATIONS PER CYCLE -1 ZONE 5:7
```

Note that reversing the order of the keywords would not work because resetting the default will override the initial selection.

10.3 Keyword Details

The following is an alphabetical list of all the WIND keywords, with detailed descriptions of their syntax and usage. Every attempt was made to build keyword inputs from intuitive, English-language words, and to adhere to some general rules of construction for the keyword commands in the data file.

ACCELERATE — Convergence acceleration (block)

```
ACCELERATE
   ZONE n
   [SECOND START iter1 END iter2]
   [FOURTH START iter1 END iter2]
   [CFLRAMP START iter1 END iter2]
ENDACCELERATE
```

The objective of WIND's convergence acceleration scheme is to approach the steady state solution more quickly by allowing the use of a large CFL number early in the calculation, adding second- and fourth- order smoothing to maintain numerical stability. This option must be used in conjunction with the SMOOTHING keyword and TEST 49.

This keyword does not work when CFL# MODE 3 (i.e., the time step calculation procedure from OVERFLOW) is used.

$iter_1$ and $iter_2$ specify the starting and ending iteration values for the corresponding keyword parameter. The CFL number will be ramped down linearly from its starting value, specified using the CFL# keyword, to 1.0.

The ZONE keyword identifies the zone within which convergence acceleration is to be used. If convergence acceleration is to be used in multiple zones, and/or if different starting and ending iteration values are to be used in different zones, multiple ZONE keywords must be specified.

If starting and ending iteration values are not specified, the following default values are used:

Option	$iter_1$	$iter_2$
SECOND	1	150
FOURTH	50	500
CFLRAMP	50	500

Example

In the following example, convergence acceleration is applied in zones 3 and 4. Second-order smoothing is applied between iterations 1 and 200, and fourth-order smoothing is applied between the default values of iterations 50 and 500. The CFL number specified with the CFL# keyword will be used for the first 250 iterations, then ramped down to a value of 1.0 between iterations 250 and 500.

```
SMOOTHING SECOND s1 FOURTH s2 SMLIMT 0.
ACCELERATE
   ZONE 3
   ZONE 4
   SECOND START 1 END 200
   CFLRAMP START 250 END 500
ENDACCELERATE
TEST 49 2
```

Recommended values for the smoothing parameters s_1, s_2, and for the starting CFL number, are:

Dimensions	s_1	s_2	CFL #
2D	0.1	0.03	10–15
3D	0.06	0.01	2.5

See Also: SMOOTHING, CFL#, TEST 49

ACTUATOR | SCREEN — Discontinuous change across a zone boundary (block)

```
{ACTUATOR | SCREEN}
   ZONE iz1 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
        [SUBSET I range J range K range]
   ZONE iz2 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
        [SUBSET I range J range K range]
   TURNING {CONSERVE {ANGLE | PARALLELU} | \
            ZERO PARALLELU | \
            {SOLIDBODY | VORTEX} val x_c y_c z_c | \
            SPECIFY ANGLE α [ROTATE β]}
   TIP-EFFECT r1 r2 r3 r4
   POWER {{DPS | DPT | DPOWER} val | \
          TURNING | \
          {SOLIDBODY | VORTEX} val x_c y_c z_c}
   EFFICIENCY {ETA val | \
               CLOSS val | \
               VORTEX val | \
               SCREEN {NORMAL | TOTAL} SOLIDITY sol}
{ENDACTUATOR | ENDSCREEN}
```

This keyword enables the user to specify a discontinuous change in properties across a zone boundary or portion of a zone boundary. Examples include actuator disks, engine face models, and screens.

The following restrictions apply:

- The `BOUNDARY TVD FACTOR 0` keyword option should be used for all actuator disk and screen boundaries.
- Screens require zero work (`POWER DPOWER 0.`)
- Only one storage location for center of rotation, the last one encountered is used for all centers.

The various elements of the `ACTUATOR | SCREEN` input block are defined as follows:

```
{ACTUATOR | SCREEN}
```

Defines the beginning of the actuator or screen block.

```
ZONE iz1 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
     [SUBSET I range J range K range]
ZONE iz2 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
     [SUBSET I range J range K range]
```

These two lines define the location of the actuator disk or screen. The relevant zones are given by the values of *iz1* and *iz2*, and the relevant boundaries within zones *iz1* and *iz2* are specified via the `BOUNDARY` keyword parameter.

iz1 Zone to which increments will be added when passing information to *iz2*

iz2 Zone receiving positive increments, increments will be subtracted when passing information back to zone *iz1*

The `SUBSET` parameter may be used to specify that the change in properties occurs only over a part of the zone boundary. Otherwise, it is assumed that the change occurs over the entire boundary. The *range* parameters define the part of the zone boundary over which the change occurs, and take one of the following forms:

index1 index2 Starting and ending indices in the specified direction. `LAST` may be used for the last index.

`ALL` Equivalent to `1 LAST`.

The starting and ending indices for the appropriate `I`, `J`, or `K` parameter (depending on the boundary specified) must be the same, and correspond to that boundary. In addition, for two-dimensional cases, the `K` parameter must be specified as either `K 1 1` or `K ALL`.

```
TURNING {CONSERVE {ANGLE | PARALLELU} | \
         ZERO PARALLELU | \
         {SOLIDBODY | VORTEX} val x_c y_c z_c | \
         SPECIFY ANGLE α [ROTATE β]}
```

Defines the net change in parallel velocity across the zone boundary.

`CONSERVE` Conserves the net flow angle (`ANGLE`) or the parallel velocity components (`PARALLELU`) across the zone boundary. (The `ANGLE` option is currently not implemented.)

`ZERO PARALLELU` Sets the parallel components of velocity across the zone boundary to zero

`SOLIDBODY` Defines a solidbody rotation increment to the parallel velocity, where:

- *val* Rotation rate in radians/second (positive by right hand rule)
- x_c, y_c, z_c Center of rotation (must be in the plane, requires the zone boundary to lie in a constant x, y, or z plane) (inches)

`VORTEX` Defines free vortex flow increment to parallel velocity, where:

- *val* Vortex strength κ (ft^2/sec), where $\kappa = \omega a^2 = \Gamma/2\pi$, and Γ is the circulation, ω is the rotation rate of the solidbody core, and a is the radius of the solidbody core (required to avoid $P = 0$ at axis), $a^2 = \kappa^2\rho/(0.9P_\infty)$ (assumes $P_{min} = 0.1P_\infty$)
- x_c, y_c, z_c Center of rotation (inches)

`SPECIFY ANGLE` Allows the user to specify the flow turning angle.

- *alpha* The flow angle giving the rotation of the *iz2* boundary normal, projected onto the x-y plane, about the z-axis (degrees)
- *beta* An optional rotation of the resulting vector about the y-axis

```
TIP-EFFECT r1 r2 r3 r4
```

Forces increments to go to zero at hub and/or tip to avoid solution discontinuities at the boundaries. A scalar, (0–1) multiplies the turning and power when this option is on. This is required for engine face models (where the wall velocity at the tip must be zero in the diffuser frame of reference). *r1–r4* define linear regions ranging from 0 to 1 between *r1* and *r2*, and from 1 to 0 between *r3* and *r4*. *r1*, *r2*, *r3*, and *r4* define the distance from the center of rotation (inches).

This keyword requires that `TURNING SOLIDBODY`, `TURNING VORTEX`, or `POWER SOLIDBODY` be specified.

```
POWER {{DPS | DPT | DPOWER} val | \
       TURNING | \
       {SOLIDBODY | VORTEX} val x_c y_c z_c}
```

Defines the power increment across the zone boundary. Screens require setting the power to zero. i.e., `POWER DPOWER 0.`

`DPS` — *val* specifies the static pressure increment across the actuator boundary (psi). Requires that the efficiency be specified, using `EFFICIENCY ETA`.

`DPT` — *val* specifies the total pressure increment across the actuator boundary (psi). Requires that the efficiency be specified, using `EFFICIENCY ETA`.

`DPOWER` — *val* specifies a (constant) power per unit area increment. (ft-lb/sec-ft^2). (Corresponds to unsteady (rotor) free vortex turning)

`DPOWER` $= \rho u c_p (T_{t2} - T_{t1})$

`TURNING` — Specifies work corresponding to the net turn across the zone boundary (specified in the `TURNING` element). Assumes all turning is done in an unsteady process (by the rotor), i.e., no stator.

$dW = c_p(T_{t2} - T_{t1}) = \omega r(w_2 - w_1)$

where ω is the rotation rate of the rotor, r is the local radius from the center of the rotor, and w is the local circumferential velocity.

This option requires that `TURNING SOLIDBODY` be specified.

`SOLIDBODY | VORTEX` — Defined as in the `TURNING` element. This defines the turning accomplished by the rotor. The net turning may be altered by another process (e.g., by a stator).

$dW = c_p(T_{t2} - T_{t1}) = \omega r(w_2 - w_1)$

Note: Currently vortex turning (i.e., `POWER VORTEX`) is not allowed. This would correspond to constant work across the rotor. However, currently, the procedure used to eliminate the vacuum at the core (setting $P_{min} = 0.1P_\infty$) makes the work input independent of the strength of the vortex, so the user could not vary the work input by changing κ.

```
EFFICIENCY {ETA val | \
            CLOSS val | \
            VORTEX val | \
            SCREEN {NORMAL | TOTAL} SOLIDITY sol}
```

Defines the efficiency of the actuator disk or screen.

`ETA` Compressor efficiency, $val = [(P_{t2}/P_{t1})^{(\gamma-1)/\gamma} - 1]/[(T_{t2}/T_{t1}) - 1]$

`CLOSS` Loss coefficient, $val = (P_{t1} - P_{t2})/q$, where $q = \rho U^2/2$ (U based on normal Mach number)

`VORTEX` Free vortex total pressure loss. The value *val* is the maximum value of $(P_{t2} - P_{t1})/P_{tinf}$ (i.e., the loss at the center of the vortex). A linear distribution is assumed from the vortex center to the radius a, where a is determined by the strength value specified using `TURNING VORTEX`, or directly using `TEST 180`.

This option requires that `TURNING VORTEX` be specified.

`SCREEN` Use screen loss relations to define total pressure loss, where

- `NORMAL` Use normal component of Mach number
- `TOTAL` Use total Mach number. (This option is not currently implemented.)
- *sol* Solidity of screen $= A_b/(A_b + A_o)$, where A_b is the blocked area, and A_o is the open area.

If solidity is specified, the screen loss coefficient associated with the screen model is defined by the solidity correlation of Cornell (1958), unless the optional `CLOSS` value is specified.

The screen model is not intended for use with choked screens, where the screen is significantly limiting the mass flow rate. During the solution start-up phase, it may be necessary to specify a low solidity, then increase it to the desired value to avoid strong choking in transients.

This option requires that the power be zero. i.e., `POWER DPOWER 0.`

`{ENDACTUATOR | ENDSCREEN}`

Ends actuator or screen input block

Examples

The following examples illustrate the use of the `ACTUATOR | SCREEN` input block for an engine face and for a screen.

Engine face model

```
ACTUATOR
   ZONE 1 BOUNDARY IMAX
   ZONE 2 BOUNDARY I1
   TURNING SOLIDBODY 240000. 312. 54. 0.
   TIP-EFFECT 5. 5.1 39.8 40.0
   POWER TURNING
   EFFICIENCY ETA 0.85
ENDACTUATOR
BOUNDARY TVD FACTOR 0 ZONE 1 BOUNDARY IMAX
BOUNDARY TVD FACTOR 0 ZONE 2 BOUNDARY I1
```

Screen

```
SCREEN
   ZONE 3 BOUNDARY K1
   ZONE 2 BOUNDARY IMAX
   TURNING ZERO PARALLELU
   POWER DPOWER 0.0
   EFFICIENCY SCREEN NORMAL SOLIDITY 0.1
ENDSCREEN
BOUNDARY TVD FACTOR 0 ZONE 3 BOUNDARY K1
BOUNDARY TVD FACTOR 0 ZONE 2 BOUNDARY IMAX
```

See Also: BOUNDARY TVD, TEST 180

ARBITRARY INFLOW — Arbitrary inflow (block)

```
{ARBITRARY INFLOW | DIFFUSER INFLOW}
   [STATIC | TOTAL]
   [HOLD_TOTALS | HOLD_CHARACTERISTICS]
   [DIRECTION {SPECIFIED | NORMAL [TO INFLOW PLANE] | ALONG [GRID LINES]}]
   ZONE n
   [UNIFORM [M P T α β] [val_k [val_om]]
      [sp1 sp2 ...spn]]
   [IJK_RANGE [FROZEN] imin imax jmin jmax kmin kmax M P T α β [val_k [val_om]]
      [sp1 sp2 ...spn]]
   [UNSTEADY var_name freq ampl phase]
   [{VORTEX | SOLIDBODY | ROTATESOLID} Mn P T α β xc yc zc \
      {dw1 | dw1 dw2 dw3}]
   [USERSPEC fs bl1 bl2 npts
      y1 M P T α β
      y2 M P T α β
      ...
      yn M P T α β]
   [USERCHEM fs bl1 bl2 npts
      y1 M P T α β sp1 sp2 ...spn
      y2 M P T α β sp1 sp2 ...spn
      ...
      yn M P T α β sp1 sp2 ...spn]
   [USERKE fs bl1 bl2 npts
      y1 M P T α β val_k [val_om]
      y2 M P T α β val_k [val_om]
      ...
      yn M P T α β val_k [val_om]]
   [USERCHEMKE fs bl1 bl2 npts
      y1 M P T α β sp1 sp2 ...spn val_k [val_om]
      y2 M P T α β sp1 sp2 ...spn val_k [val_om]
      ...
      yn M P T α β sp1 sp2 ...spn val_k [val_om]]
[ENDINFLOW]
```

Several options are available to set conditions at arbitrary inflow boundaries. The default is uniform inflow (i.e., no boundary layer) at the conditions that are set using the FREESTREAM keyword. Other options are selected by using either of the equivalent keywords ARBITRARY INFLOW or DIFFUSER INFLOW.

The remaining lines select the specific type of nonuniform entrance data to be provided. Discussion of the input data for each of these options is presented in the remainder of this subsection, grouped by "Control Functions", "Condition Specification", and "Special Capabilities". These keywords can start in any column. Generally, they should be indented from the ARBITRARY INFLOW keyword to set them apart.

This keyword may also be used to initialize (or reinitialize) the flow conditions within the specified zone, as described in Section 3.6.

Note: The GAS keyword and the CHEMISTRY keyword block, if used, must come before the ARBITRARY INFLOW keyword block in the input data (*.dat*) file.

Control Functions

`ENDINFLOW`

This optional keyword may be used to end the arbitrary inflow definition.

`STATIC | TOTAL`

Arbitrary inflow conditions specified after this keyword will be considered as static or total, depending which of these is set. By default, the flag is set to the value from the FREESTREAM keyword.

Note: `TOTAL` is only valid for an ideal gas.

`HOLD_TOTALS | HOLD_CHARACTERISTICS`

Specifying `HOLD_TOTALS` indicates that total temperature and local flow angles are to be held at their specified values, and is only valid for an ideal gas. Specifying `HOLD_CHARACTERISTICS` indicates that characteristic values are to be held constant. The option specified will be applied at all the arbitrary inflow regions in the zone, and will remain in effect for all following `ZONE` keywords.

In WIND 5.201 and later the default is `HOLD_CHARACTERISTICS`. In earlier versions the default is `HOLD_TOTALS`.

`HOLD_TOTALS` only works in conjunction with the `UNIFORM`, `IJK_RANGE`, `VORTEX`, `SOLIDBODY`, and `ROTATESOLID` keywords.

The total pressure is always held fixed, whether `HOLD_TOTALS` is specified or not. Although the Mach number is specified with the `UNIFORM` and `IJK_RANGE` keyword parameters, it may be adjusted during the boundary condition treatment.

Note that the `HOLD_TOTALS` keyword in the `ARBITRARY INFLOW` keyword block applies to arbitrary inflow boundaries only. See the HOLD keyword for information on holding conditions at freestream boundaries with inflow.

Note also that the syntax is slightly different for arbitrary inflow and freestream boundaries. For arbitrary inflow boundaries, `HOLD_TOTALS` and `HOLD_CHARACTERISTICS` are used in the `ARBITRARY INFLOW` keyword block, with an underscore. For freestream boundaries, `HOLD TOTALS` and `HOLD CHARACTERISTICS` are used, without an underscore.

`DIRECTION {SPECIFIED | NORMAL [TO INFLOW PLANE] | ALONG [GRID LINES]}`

The `DIRECTION` keyword indicates how the flow angle is to be set at an inflow plane. The options for setting the flow direction are:

`SPECIFIED`	Set the flow at the angles of attack and yaw specified elsewhere in the `ARBITRARY INFLOW` block. This is the default.
`NORMAL`	Set the flow normal to the inflow plane.
`ALONG`	Set the flow in the direction of the grid lines intersecting the inflow plane.

Specifying `DIRECTION NORMAL` or `DIRECTION ALONG` will override any angles of attack or yaw specified with the `UNIFORM`, `IJK_RANGE`, `VORTEX`, `SOLIDBODY`, and `ROTATESOLID` keywords. However,

if UNIFORM is used without specifying the flow conditions, the angles specified with the FREESTREAM keyword will be used, and the DIRECTION keyword will have no effect. The DIRECTION keyword also does not affect flow angles in profiles specified with the USERSPEC, USERCHEM, USERKE, or USERCHEMKE keywords.

The DIRECTION keyword will not modify the flow angles that are set when ARBITRARY INFLOW is being used to initialize (or reinitialize) flow conditions within a zone.

The DIRECTION option used will remain in effect for all following ZONE keywords.

ZONE n

This keyword, which *must* be specified, identifies the zone for which inflow conditions are being set. For example, if zone 2 is an internal jet, conditions other than freestream may be desired at the inflow to zone 2.

Within the ARBITRARY INFLOW keyword block, the ZONE keyword must come *before* any "Condition Specification" keyword for that zone, but *after* any HOLD_TOTALS, HOLD_CHARACTERISTICS, or DIRECTION keyword for that zone. In addition, inflow conditions may only be specified for one zone at a time.

Example

The following ARBITRARY INFLOW block specifies that total conditions are to be held constant at arbitrary inflow surfaces in zones 1 and 2, with $M = 3.5$, $P_T = 251.15$ psi, and $T_T = 1167.9$ °R. In zone 3, characteristic values are to be held constant at arbitrary inflow surfaces, consistent with the flow conditions given with the FREESTREAM keyword.

```
ARBITRARY INFLOW
   TOTAL
   HOLD_TOTALS
      ZONE 1
      UNIFORM 2.5 251.15 1167.9 0.0 0.0
      ZONE 2
      UNIFORM 2.5 251.15 1167.9 0.0 0.0
   HOLD_CHARACTERISTICS
      ZONE 3
      UNIFORM
ENDINFLOW
```

Condition Specification

UNIFORM [M P T α β] [val_k [val_om]]
[sp_1 sp_2 ...sp_n]

The entrance flow is uniform, at the conditions specified. If no flow conditions are specified, those specified with the FREESTREAM keyword are used No other data is provided to select this option. This is the default option, and is selected automatically if no extended input options are selected.

Different conditions can be specified within each zone. This is done by adding the run conditions after the keyword. These data are added in free format, one or more blanks separating values.

M Mach number

P	Pressure
T	Temperature
α, β	Angles of attack and yaw, relative to the Cartesian x direction
sp_1, sp_2, ..., sp_n	Species mass fractions

Pressure and temperature are static or total, depending on whether `STATIC` or `TOTAL` is specified. As always, pressure is specified in pounds per square inch, temperature in degrees Rankine, and α and β in degrees.

If the SST turbulence model is being used (see the `TURBULENCE` keyword), *val_k* and *val_om* may be used to specify inflow turbulence levels. Note that you may specify either *val_k*, or *val_k* and *val_om*, but not *val_om* by itself. Note also that if these values are being specified, the `TURBULENCE` keyword must come before the `ARBITRARY INFLOW` keyword block in the input data (*.dat*) file.

The following options are possible:

val_k > 0 The turbulent kinetic energy k and the specific dissipation rate ω are specified directly, with

$$k = val_k \ (\mathrm{ft}^2/\mathrm{sec}^2)$$
$$\omega = val_om \ (1/\mathrm{sec})$$

The turbulent viscosity ν_t is then equal to k/ω.

val_k < 0 The turbulent kinetic energy k is set equal to *val_k* percent of the "reference" kinetic energy $U^2/2$, where U is computed from the specified values of M and T. Thus

$$k = 0.01 \, |val_k| \, \frac{U^2}{2}$$

The turbulent viscosity ν_t is automatically set equal to $0.001\nu_l$, where ν_l is the laminar viscosity, and the specific dissipation rate is computed as $\omega = k/\nu_t$.

val_om < 0 The specific dissipation rate ω is set equal to *val_om* percent of U/L_{ref}, where U is computed from the specified values of M and T, and L_{ref} is the reference length from the grid (*.cgd*) file. Thus

$$\omega = 0.01 \, |val_om| \, \frac{U}{L_{ref}}$$

The turbulent viscosity ν_t is set to the same percentage of the laminar viscosity.

$$\nu_t = 0.01 \, |val_om| \, \nu_l$$

The turbulent kinetic energy is then computed as $k = \omega\nu_t$.

If inflow turbulence levels are not specified using one of the above options, or if *val_k* $= 0$, default values are computed from

$$\omega = 10U/L_{ref}$$
$$\nu_t = 0.001\nu_l$$
$$k = \omega\nu_t$$

Note that

- If *val_k* > 0, a positive value must be specified for *val_om*.
- If *val_k* ≤ 0, *val_om* should not be specified.
- If *val_om* < 0, a value must also be specified for *val_k*, but it is ignored.

Example

```
ARBITRARY INFLOW
   UNIFORM 1.1 100. 900. 10. 0.
ENDINFLOW
```

`IJK_RANGE [FROZEN]` *imin imax jmin jmax kmin kmax M P T* α β `[`*val_k* `[`*val_om*`]]` `[`sp_1 sp_2 $\ldots sp_n$`]]`

This keyword allows specification of inflow conditions over an arbitrary range of i, j, and k indices on any computational boundary plane. The user specifies the minimum and maximum i, j, and k indices which describe the region, followed by the flow conditions to be applied, as follows:

imin, imax	Minimum and maximum i indices bounding the region
jmin, jmax	Minimum and maximum j indices bounding the region
kmin, kmax	Minimum and maximum k indices bounding the region
M	Mach number
P	Pressure
T	Temperature
α, β	Angles of attack and yaw, relative to the Cartesian x direction
sp_1, sp_2, $\ldots$, sp_n	Species mass fractions

Pressure and temperature are static or total, depending on whether `STATIC` or `TOTAL` is specified. As always, pressure is specified in pounds per square inch, temperature in degrees Rankine, and α and β in degrees. There are no defaults for the index ranges.

The `FROZEN` option may be specified to freeze the inflow conditions over the indicated index range at their current values. Note that for a restart case (i.e., when a *.cfl* file already exists), the "current values" are those in the *.cfl* file, not those specified with the `IJK_RANGE` parameter. For an initial run (i.e., when a *.cfl* file does not exist), the flow conditions will be frozen at the conditions specified with the `IJK_RANGE` parameter.

If the SST turbulence model is being used (see the `TURBULENCE` keyword), *val_k* and *val_om* may be used to specify inflow turbulence levels. The various options are described above under the `UNIFORM` keyword.

Up to 500 `IJK_RANGE` keywords may be specified. This is useful when specifying a boundary layer profile at an inflow boundary, or along solid walls during the flowfield initialization process, as described in Section 3.6 starting on p. 36.

`UNSTEADY` *var_name freq ampl phase*

This keyword allows the user to specify unsteady arbitrary inflow conditions, and it must be used with (and follow) the `IJK_RANGE` keyword. Up to three different perturbations to the inflow conditions may be specified and will be superimposed to create unsteadiness centered about the conditions given with the `IJK_RANGE` keyword.

var_name	One of the keywords `MACH`, `PRESSURE`, `TEMPERATURE`, `ALPHA`, `BETA`, or `VELOCITY`
freq	Frequency of the perturbation in Hertz
ampl	Amplitude of the perturbation in appropriate variable units
phase	Phase angle of the perturbation in degrees

Note that you may specify multiple, independent pairs of `IJK_RANGE` and `UNSTEADY` keywords.

`{VORTEX | SOLIDBODY | ROTATESOLID}` M_n P T α β *xc yc zc* `{`*dw1* `|` *dw1 dw2 dw3*`}`

Specifies uniform inflow conditions with free-vortex or solid-body rotation superimposed. Solid-body rotation may be specified on any arbitrary inflow boundary. For free-vortex rotation, however, the arbitrary inflow boundary must be a constant x, y, or z plane, and the center of rotation must lie on that plane.

M_n	Normal component of Mach number
P	Pressure (psia)
T	Temperature (°R)
α, β	Average angles of attack and yaw, relative to the Cartesian x direction
xc, yc, zc	Center of rotation in physical coordinates

For `VORTEX`,

dw1	Vortex strength. (See the `ACTUATOR` keyword.)

For `SOLIDBODY` and `ROTATESOLID`,

dw1, dw2, dw3	x, y, and z components of the rotation rate vector (radians/sec)

The rotational velocity components are added in such a way that total pressure and total temperature are held constant at the inflow boundary. Thus, the `TOTAL` option should always be used for this mode, since with the `STATIC` option the computed static pressure and temperature at the inflow boundary may differ from the specified values.

For calculations in a rotating reference frame (see the `ROTATE` keyword):

- If the `SOLIDBODY` option is used, the total conditions specified are those in the rotating frame, and are held fixed in that frame. The total conditions in the inertial frame will vary.

- If the ROTATESOLID option is used, the total conditions specified are those in the inertial frame, and are held fixed in that frame. The total conditions in the rotating frame will vary. Note that the ROTATESOLID option is only valid for calculations in a rotating reference frame.

For both free-vortex and solid-body rotation, the flowfield must already be initialized. (See Section 3.6.) I.e., there must be a pre-existing *.cfl* file. The VORTEX, SOLIDBODY, and ROTATESOLID options cannot be used during a "cold" start.

Special Capabilities

The following arbitrary inflow keywords effect changes only to the $i = 1$ computational plane.

```
USERSPEC fs bl1 bl2 npts
   y1 M P T α β
   y2 M P T α β
   ...
   yn M P T α β
```

This option allows the user to specify a 1-D profile normal to the surface, translated through some buttline range, below the vehicle. These conditions will be set last and thus the data will overwrite UNIFORM conditions over the range of interest.

fs	Fuselage station of the profile (to be checked against the grid $i = 1$ fuselage station)
bl1, bl2	Minimum and maximum buttline over which to translate the profile
npts	Number of points defining the profile
y1 – yn	Normal distance from the wall
M	Mach number
P	Pressure (psia)
T	Temperature (°R)
α, β	Angles of attack and yaw, relative to the Cartesian x direction

The pressure and temperature may be the total or static conditions, depending upon the current setting of the TOTAL/STATIC keyword. If neither STATIC nor TOTAL have been specified under ARBITRARY INFLOW, then the existing switch from the global input parameters is used (default: TOTAL).

One profile can be specified for each zone. There can be 100 points in each profile. The normal distance is always assumed to be from $j = 1$ (the reference wall is assumed to be at $j = 1$). *bl1* is the minimum buttline and *bl2* is the maximum buttline.

See Also: TEST 157
By default, USERSPEC only specifies conditions below a vehicle. That is, the wall ($j = 1$) must be above (higher y) the interior grid points. TEST 157 specifies that all points within the specified buttline range will be affected, above and below the vehicle. This should be the default, but isn't.

```
USERCHEM fs bl1 bl2 npts
   y1 M P T α β sp1 sp2 ...spn
   y2 M P T α β sp1 sp2 ...spn
   ...
   yn M P T α β sp1 sp2 ...spn
```

The USERCHEM option is identical to the USERSPEC option, except that chemistry species mass fractions sp_1, sp_2, ..., sp_n are added to the end of each specified profile. Test options can then be set to model the mixing of gas streams which have different chemical compositions. At this time, only mixing can be modeled. The gas streams cannot chemically react.

Note: Only the STATIC input mode is available for chemistry.

Note: For TEST 75 1, the *sp* input parameters specify the mass fractions of O_2, CO_2, H_2O, NO_2, and N_2, in that order. For TEST 75 2, they specify the mass fractions of O_2, H, H_2, H_2O, and N_2. The user must be careful to enter the mass fractions in the order shown, entering zeroes to leave out particular species.

Part of an example USERCHEM input block follows. The file sets up a rectangular jet where the jet composition is a mixture of O_2, CO_2, H_2O, NO_2, and N_2. The last five columns hold the species mass fractions for the profiles. Note that the second-to-last column consists of zeroes; this is the unused position in the chemistry array which was described above for a four-species chemistry model.

```
ARBITRARY INFLOW
   ZONE 1
   USERCHEM 0.0 -10.0 10.0 6
     0.0   0.3  5.70   433.1  0.0  0.0  0.234  0.0    0.0    0.0  0.766
   123.05  0.3  5.70   433.1  0.0  0.0  0.234  0.0    0.0    0.0  0.766
   123.05  1.8  5.66  1940.0  0.0  0.0  0.096  0.120  0.048  0.0  0.736
   133.75  1.8  5.66  1940.0  0.0  0.0  0.096  0.120  0.048  0.0  0.736
   133.75  0.3  5.70   433.1  0.0  0.0  0.234  0.0    0.0    0.0  0.766
   257.0   0.3  5.70   433.1  0.0  0.0  0.234  0.0    0.0    0.0  0.766
ENDINFLOW
TEST 157 1     USERSPEC ABOVE AND BELOW VEHICLE
```

```
USERKE fs bl1 bl2 npts
   y1 M P T α β val_k [val_om]
   y2 M P T α β val_k [val_om]
   ...
   yn M P T α β val_k [val_om]
```

The USERKE option is identical to the USERSPEC option, except that the values *val_k* and *val_om* are added to specify inflow turbulence levels when the SST turbulence model is being used. The various options are described above under the UNIFORM keyword.

```
USERCHEMKE fs bl1 bl2 npts
   y1 M P T α β sp1 sp2 ...spn val_k [val_om]
   y2 M P T α β sp1 sp2 ...spn val_k [val_om]
   ...
   yn M P T α β sp1 sp2 ...spn val_k [val_om]
```

The USERCHEMKE option is identical to the USERCHEM option, except that the values *val_k* and

val_om are added to specify inflow turbulence levels when the SST turbulence model is being used. The various options are described above under the UNIFORM keyword.

See Also: TURBULENCE, REINITIALIZE, ROTATE

AXISYMMETRIC | AXI-SYM — Axisymmetric flow

`{AXISYMMETRIC | AXI-SYM}` *yc theta*

This keyword allows a two dimensional grid to be run assuming axisymmetric flow. Note that k_{max} must be 1, and that the *.cgd* file is unaware that the grid is axisymmetric or 2D; that is determined at WIND run time.

The user-specified values are defined as:

yc Waterline location of rotation axis, in input units (scaled in WIND by `XLONG*SCAFAC` and offset by `YNOSE`)

theta Degrees of rotation assumed for metrics. (Changing *theta* should not affect the solution, but will affect convergence. I like 5 degrees.)

BLEED — Bleed region flow rate

```
BLEED {ibrg blv1 | POROSITY ibrg blv1 blv2 blv3 | MODEL ibrg mode blv1 blv2 blv3 blv4 | \
       FORCING ibrg blv1 blv2 blv3}
```

The effect of bleed on the flow can be modeled, if bleed regions were identified in the grid file. The parameters discussed below identify the bleed rate for each region, for a specific solution. If a bleed region is not named in this file, its bleed rate is set to zero.

There are four possible bleed models, as follows:

`BLEED` *ibrg blv1*

ibrg Bleed region number from *.cgd* file

blv1 Normalized bleed flow rate

blv1 can also be thought of as the mass flow ratio for the bleed region. The actual bleed mass flow is calculated as

$$\dot{m}_b = blv1\,(\rho_\infty U_\infty A_c)$$

where A_c is an arbitrarily set reference "capture area" that you must specify, in either GMAN (using the `CAPTURE AREA` command, or the `BOUNDARY COND.` menu) or MADCAP (using "Set Capture Area" from the "Boundary Conditions" menu), in each zone of the grid file that contains bleed areas.

The bleed velocity will automatically be limited to Mach 1.

Although this is intended as a bleed model, it can also be used for blowing by setting *blv1* to a negative value. Note, however, that if the resulting blowing velocity exceeds Mach 1, the logic in the code that is used to limit the *bleed* velocity to Mach 1 will reset *blv1* to a positive value, resulting in bleeding instead of blowing.

`BLEED POROSITY` *ibrg blv1 blv2 blv3*

ibrg Bleed region number from *.cgd* file

blv1 Back pressure p_{plen}, in psia

blv2 Porosity

blv3 Discharge coefficient

With this model, the velocity at the wall will be computed from the local pressure p in the flow field, and the specified back pressure p_{plen}. If $p > p_{plen}$, the flow will be out of the computational domain (i.e., bleed). If $p < p_{plen}$, the flow will be into the computational domain (i.e., blowing).

`BLEED MODEL` *ibrg mode blv1 blv2 blv3 blv4*

This keyword specifies use of the empirical bleed model of Mayer and Paynter (1994).

The input parameter *ibrg* is the bleed region number from the *.cgd* file. The input data for the bleed model is given by the values of *blv1* through *blv4*. Various combinations of values may be specified, depending on the *mode*, as described below.

mode	*blv1*	*blv2*	*blv3*	*blv4*
1	p_{plen}	Porosity	*qsmode*	N_{bl}
2	p_{plen}	Porosity	*qsmode*	
3	p_{plen}	Porosity		M
4	Q_{sonic}	Porosity	*qsmode*	M

In the above table, p_{plen} is the bleed plenum static pressure, N_{bl} is the number of grid points in the boundary layer, Q_{sonic} is the sonic mass flow coefficient (described below), and M is the local Mach number at the edge of the boundary layer. The parameter *qsmode* is an integer from 1 to 3 defining how Q_{sonic} is to be computed, as follows:

1 Set $Q_{sonic} = 1$
2 Compute Q_{sonic} for 90° holes
3 Compute Q_{sonic} for 20° holes

In the Mayer-Paynter model, the bleed velocity is given by the formula

$$V_{bleed} = Q_{sonic}\,\Phi\,\Gamma\,T\,\frac{p_T}{p}\,\frac{1}{\sqrt{T_T}}$$

where Φ is the porosity, p and T are the local static pressure and temperature, and p_T and T_T are the local total pressure and temperature. "Local" means at the edge of the boundary layer at the location of the grid point within the bleed region. The parameter Γ is a function of the specific heat ratio γ.

$$\Gamma = \left(1 + \frac{\gamma - 1}{2}\right)^{-\frac{\gamma+1}{2(\gamma-1)}}$$

Central to the model is Q_{sonic}, the sonic mass flow coefficient, defined as

$$Q_{sonic} = \frac{\dot{m}_{bleed}}{\dot{m}_{max}} = f\left(\alpha, M, \frac{p_{plen}}{p_T}\right)$$

where $\dot{m}_{bleed}$ is the actual bleed flow rate and $\dot{m}_{max}$ is the maximum theoretical bleed flow rate.

Q_{sonic} is a function of the bleed hole angle α, the local Mach number M, and the ratio of the plenum pressure p_{plen} to the local pressure p. The functional relationship is in the form of tabulated experimental data for circular bleed holes at angles of 20° (McLafferty and Ranard, 1958) and 90° (Syberg and Hickox, 1972).

`BLEED FORCING` *ibrg blv1 blv2 blv3*

This mode allows an oscillating normal velocity bleed boundary condition to be specified.

ibrg Bleed region number from *.cgd* file

blv1 Amplitude of the normal velocity oscillation (ft/sec)

blv2 Frequency of the oscillation (Hz)

blv3 Phase offset of the oscillation (deg)

See Also: `BLOW`, `MASS FLOW`

BLOW — Inject vectored flow over a selected region

```
BLOW {ibreg mdot T_inj angle_inc [angle_azi] | \
      PLENUM ibreg P_t T angle | \
      VALVE ibreg P_t T_t angle | \
      SURFACE ibreg P_t T_t angle_inc angle_azi}
```

Porous wall cooling over a selected region can be simulated using this keyword. The region must be identified as a bleed region in the grid file. This option is intended for mass inflow only (i.e., $P_t >$ local P_s). It won't work well for grids that are skewed at the wall, resulting in blowing mass flow errors.

Blowing is currently implemented for j-boundaries only, and the flow must be in the $+x$ direction ($u > 0$).

Blowing may also be modeled using the first two forms of the BLEED keyword (i.e., `BLEED` and `BLEED POROSITY`).

There are four possible blowing modes, as follows:

```
BLOW ibreg mdot T_inj angle_inc [angle_azi]
```

ibreg	Bleed region number from *.cgd* file
mdot	Injected mass flow in region *ibreg* ($\mathrm{lb_m}$/sec)
T_{inj}	Static temperature of injected flow (°R)
angle_inc	Blowing inclination angle (degrees); must be > 0
angle_azi	Blowing azimuthal angle (degrees)

The blowing direction is set by the input azimuthal and inclination angles.

If *angle_azi* is not specified, *angle_inc* is interpreted as the rotation about the z axis into the flow.

If *angle_azi* is specified, starting from the projection of the x axis onto the surface, the blowing direction is determined by rotating about the surface normal by the azimuthal angle, then "up" from the surface by the inclination angle. If this results in blowing in the "upstream" (defined as negative x) direction, the azimuthal angle ψ is automatically changed to $180° - \psi$, while maintaining the specified inclination angle. This reverses the x component of the blowing velocity vector, so that blowing is in the "downstream" direction, while keeping the other components unchanged.

Note that specifying both angles, with *angle_azi* $= 0$, is not the same as specifying just *angle_inc*, except on surfaces with no curvature in the z direction.

This blowing mode may not be used for multi-species flows.

TEST option 191, mode 1, may be used to specify that the blowing angles are to be measured relative to the projection of the surface normal onto a constant z plane, rather than relative to the normal itself.

TEST option 191, mode 4, allows an earlier one-angle form of the `BLOW` keyword to be used. However, this is not recommended. Due to an implementation error, this option only considers mass

flow in the x-y plane, and therefore does not provide the requested mass flux for surfaces whose normal vector contains components in the z direction (i.e., surfaces with transverse curvature).

Important: The change from the one-angle form to the two-angle form of this keyword was made in WIND 5.204. Specifying just one angle is still allowed, but will in general give slightly different results, due to the implementation error in earlier versions described above.

`BLOW PLENUM` *ibreg* P_t T *angle*

ibreg	Bleed region number from *.cgd* file
P_t	Plenum total pressure (psi)
T	Plenum static temperature (°R)
angle	Blowing angle relative to x-y plane (degrees)

If the flowfield static pressure P_s becomes greater than the plenum total pressure P_t, the plenum total pressure will be automatically increased to $1.005P_s$ to maintain a blowing boundary condition. Setting `TEST 52` will notify the user when this occurs.

`BLOW VALVE` *ibreg* P_t T_t *angle*

ibreg	Bleed region number from *.cgd* file
P_t	Plenum total pressure (psi)
T_t	Plenum total temperature (°R)
angle	Blowing angle relative to x-y plane (degrees)

If the flowfield static pressure P_s becomes greater than the plenum total pressure P_t at any point within the blowing region, blowing will be shut off for the entire region, and the surface will be treated as a solid wall. A `*VLV*` line is written to the list output (*.lis*) file whenever the valve changes status.

With `BLOW VALVE`, the blowing region may extend to more than a boundary surface, and may also be split between zones. Note, however, that when a blowing region is split between multiple processors, the separate sub-regions act independently until the end of a cycle. If the flowfield static pressure grows large enough in one sub-region to close the valve but not in the other sub-region(s), the valve will close for the first sub-region, but not on the others until the end of the cycle. The reverse situation (i.e., opening a closed valve) may also occur. This may be prevented by running one iteration per cycle; as a practical matter, it is not expected to cause problems with the default of five iterations per cycle.

The `BLOW VALVE` keyword may not be used for multi-species flows.

`BLOW SURFACE` *ibreg* P_t T_t *angle_inc* *angle_azi*

ibreg	Bleed region number from *.cgd* file

P_t	Plenum total pressure (psi)
T_t	Plenum total temperature (°R)
angle_inc	Blowing inclination angle (degrees)
angle_azi	Blowing azimuthal angle (degrees)

With this boundary condition, blowing will occur whenever the local flowfield static pressure is less than the specified plenum total pressure. If the flowfield static pressure is greater than the plenum total pressure, the velocity normal to the wall at that point is set to zero (i.e., a solid wall with no blowing or bleed). Unlike the `BLOW VALVE` capability, with `BLOW SURFACE` blowing is turned on or off locally, on a point-by-point basis.

The blowing velocity is also constrained to subsonic values.

The blowing direction is set by the input inclination and azimuthal angles. The angles are defined in the same way as when the standard `BLOW` keyword is used to specify a constant blowing mass flow, as described above, except that the one-angle form is not available.

`TEST` option 191 may be used to specify that the blowing angles are to be measured relative to the projection of the surface normal onto a constant z plane, rather than relative to the normal itself.

`TEST` option 191 may also be used to allow surface blowing with Liu-Vinokur equilibrium air chemistry. The `BLOW SURFACE` keyword may not otherwise be used for multi-species flows.

If the `TEST 195` option is set, a message will be written in the list output (*.lis*) file whenever the blowing is turned off because the flowfield static pressure is too large. Note, however, that this is a five-line message written for each iteration and each "closed" node, and could cause the *.lis* file to become very large very quickly.

The implementation of this boundary condition is general enough that one can specify a blowing region on the upper and/or lower surface of a wing. However, one should avoid specifying a blowing region in more severe cases, such as the normal part of a backward-facing step.

See Also: `BLEED`, `TEST 52`, `TEST 191`, `TEST 195`

BL_INIT — Boundary layer initialization

`BL_INIT` *nb ixs str igrs* `[`*delt*`]` *nbli nzone*

The `BL_INIT` keyword signifies the user wishes to initialize a j or k boundary with a laminar or turbulent flat plate boundary layer. The user-specified values are defined as:

nb A number from 1 to 4 indicating which boundary to initialize

- 1 for $j = 1$
- 2 for $j = j_{max}$
- 3 for $k = 1$
- 4 for $k = k_{max}$

ixs Integer indicating method for specifying the axial location to start the boundary layer:

- 1 to specify the i index
- 2 to specify the x station

str i index or x location to start the boundary layer, depending on the value of *ixs*

igrs Integer indicating method for specifying the initial height of the boundary layer:

- 1 to specify the number of grid points from the wall
- 2 to specify the thickness
- 3 to compute thickness from initial x station

delt Number of grid points in boundary layer (if $igrs = 1$), or boundary layer thickness (inches) (if $igrs = 2$)

nbli Integer indicating type of boundary layer and whether to keep constant thickness or grow with x

- 1 Turbulent profile, growing with x
- -1 Turbulent profile, constant thickness
- 2 Laminar profile, growing with x
- -2 Laminar profile, constant thickness

nzone Zone number for initialization

- 0 All zones
- > 0 Zone *nzone* only

A flat plate temperature distribution will also be constructed consistent with the wall temperature boundary condition of either adiabatic or constant temperature wall.

Note: `BL_INIT` cannot currently do more than one boundary in each zone without over-writing itself.

BOUNDARY-DAMP | BDAMP — Boundary damping (block)

```
{BOUNDARY-DAMP | BDAMP}
   ZONE range1[,range2[, ...  rangen]]
   BOUNDARY bnd1:bnd2
   [POINTS n]
   [SECOND c2]
   [FOURTH c4]
   [NOSMOOTHING [NI m1] [NJ m2] [NK m3]]
{END-BOUNDARY-DAMP | ENDBDAMP}
```

This keyword instructs WIND to utilize second- and fourth-order smoothing to damp waves in the vicinity of computational boundaries. It is also possible to turn off smoothing near any boundary for any numerical operator (ξ, η, or ζ). This option must be used with the SMOOTHING keyword and TEST 49 2. The user-specified values are defined as:

range Zone specification range, in one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
begzone:	Selects all zones from *begzone* to MAXZONE, the total number of zones in the grid file
:*endzone*	Selects all zones from 1 to *endzone*
ALL	Selects all zones

bnd1 Starting boundary (0 to begin at boundary 1). The values 1–6 correspond to the $i = 1$, $i = i_{max}$, $j = 1$, $j = j_{max}$, $k = 1$, and $k = k_{max}$ boundaries, respectively.

bnd2 Ending boundary (0 to end at last boundary)

n Number of points into the domain over which boundary damping is to be applied

c2 Second-order smoothing coefficient

c4 Fourth-order smoothing coefficient

The NOSMOOTHING keyword is used to turn off smoothing (as specified using the SMOOTHING keyword) for the most recently specified ZONE and BOUNDARY. The keywords NI, NJ, and/or NK specify that smoothing is to be turned off for the ξ, η, and/or ζ operator, respectively, for boundaries *bnd1* through *bnd2*.

m_1, m_2, m_3 Number of points into the domain on boundaries *bnd1* through *bnd2* over which smoothing is to be turned off

Example

```
BOUNDARY-DAMP
   ZONE 2:4 BOUNDARY 1:3 POINTS 15 SECOND .01 FOURTH .03
   ZONE 3:3 BOUNDARY 4:4 POINTS 10 SECOND .02 FOURTH .01
   ZONE 5:5
   BOUNDARY 2:2
   NOSMOOTHING NI 31 NJ 21
END-BOUNDARY-DAMP
```

See Also: SMOOTHING, TEST 49

BOUNDARY TVD — Boundary total variation diminishing operator flag

```
BOUNDARY TVD [{OFF | FACTOR factor} [ZONE range1[,range2[, ...  rangen]] \
   [BOUNDARY {ALL | I1 | IMAX | J1 | JMAX | K1 | KMAX | OVERLAP}]]]
```

This keyword controls the TVD flux limiter in the explicit operator at the specified coupled boundaries. The same type of limiter (minmod, Koren, or van Albada) will be used as for the internal scheme.

`FACTOR` specifies the "compression" parameter for the TVD operator at the specified boundaries, which controls the amount of limiting. Larger numbers result in less limiting towards a first order operator. Setting *factor* to zero is equivalent to a first order operator, one corresponds to no overshoots, and two allows some increase in interface value over surrounding values.

The boundary TVD limiter must be more limiting than the interior TVD limiter for it to have any additional effect. I.e., the value *factor* must be less than the value used for the internal scheme.

In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone`:`*endzone*	Selects all zones from *begzone* to *endzone*
begzone`:`	Selects all zones from *begzone* to `MAXZONE`, the total number of zones in the grid file
`:`*endzone*	Selects all zones from 1 to *endzone*
`ALL`	Selects all zones

By default, boundary TVD is on at all boundaries. If the `BOUNDARY TVD` keyword is not specified at all, the compression parameter will be the same as is used for the internal scheme. If `BOUNDARY TVD` is specified, but without additional keywords, the default value for *factor* will be used, as listed for the `TVD` keyword.

Note that if a particular zone (or zones) is to be specified, then either "`OFF`" or "`FACTOR` *factor*" must also be specified. And, if a particular boundary is to be specified, then the zone(s) must also be explicitly specified.

See Also: `TVD`

CFL# — CFL/time step specification

```
{CFL# | TIMESTEP} { \
   [MODE 1] [CFL | SECONDS] {cfl [zone_selector [ityp]] | \
   [MODE 2] [CFL | SECONDS] INCREMENT cfl1 cflmax cflfac inccfl istart} \
      [zone_selector [ityp]] | \
   MODE 3 [DT dt] [CFLMIN cflmin] [CFLMAX cflmax] [ITIME itime] [zone_selector]}
```

This keyword allows the user to specify the CFL number or time step for all zones, or on a zone-by-zone basis. All keywords and parameters must be on one line in the data file. If this keyword is not used, a constant CFL number of 1.3 is used, equivalent to specifying "`CFL# CFL 1.3`".

There are three different options, as specified by the `MODE` value. Note that the `MODE` keyword is optional for modes 1 and 2, but required for mode 3. (Modes 1 and 2 are distinguished in the code by whether or not the word `INCREMENT` is present.)

```
{CFL# | TIMESTEP} [MODE 1] [CFL | SECONDS] {cfl [zone_selector [ityp]]
```

In this mode, the CFL number or time step is specified directly, by the input value of *cfl*.

`CFL` A CFL number is being specified. This is the default, and should be used for steady flow problems.

`SECONDS` The time step is being specified in seconds. This option is generally used for unsteady flow problems.

When a CFL number is being specified, the value of *ityp* may be set to 1 to indicate that a global time step (i.e., constant in space) should be used, equal to the minimum value in the zone. This allows the time step to be determined through a CFL number for unsteady flow problems. Currently, this option is only available for single-zone grids. When a CFL number is specified, and $ityp = 0$ or is omitted, a local time step (i.e., varying in space) will be used.

```
{CFL# | TIMESTEP} [MODE 2] [CFL | SECONDS] INCREMENT cfl1 cflmax cflfac inccfl istart} \
   [zone_selector [ityp]]
```

In this mode, the CFL number or time step will be gradually increased as the calculation proceeds. The `CFL` and `SECONDS` options have the same meaning as for mode 1, again with `CFL` as the default. The increase in the CFL number or time step is controlled by the following input values:

cfl1 CFL number or time step on first iteration

cflmax Maximum CFL number or time step allowed

cflfac Factor which multiplies the CFL number or time step every *inccfl* iterations

inccfl Iteration increment at which the CFL number or time step is multiplied by *cflfac*. The multiplication is done whenever mod (iteration, *inccfl*) = 1.

istart Starting iteration in the increment calculation

As in mode 1, when a CFL number is being specified for a single-zone grid, the value of *ityp* may be set to 1 to indicate that a global time step (i.e., constant in space) should be used, equal to the minimum value in the zone.

```
{CFL# | TIMESTEP} MODE 3 [DT dt] [CFLMIN cflmin] [CFLMAX cflmax] [ITIME itime] \
   [zone_selector]
```

This mode uses the time step calculation procedure originally used in the OVERFLOW code.

itime A flag indicating the type of time step to be used

- 0 Time-accurate run (i.e., time step is constant in space)
- 1 Time step scaled by local metric Jacobian, with fudge factor of 0.005 to keep the time step from getting too small for really tiny cells
- 2 Time step scaled by local metric Jacobian, without a fudge factor
- 3 Constant CFL number equal to *cflmax*

dt Time step parameter. For *itime* = 0 (i.e., time-accurate cases), this is the time step in seconds nondimensionalized by L_r/a_r, where L_r is the grid reference length and a_r is the freestream speed of sound. For *itime* = 1 and 2, this is a CFL number, before scaling by the local metric Jacobian. For *itime* = 3, *dt* is not used. The default value is 0.5.

cflmin, cflmax For *itime* = 3, the CFL number is set to *cflmax*, and *cflmin* is not used. The CFL number is defined using the sum of the maximum eigenvalues in each coordinate direction.

For other *itime* values, *cflmin* and *cflmax* are used to limit the CFL number to values within the specified range. A value of 0.0 indicates no limiting. If either *cflmin* or *cflmax* is negative, the absolute values are used, and the CFL number is defined using the method of Gnoffo, with a viscous correction by Tannehill. If both *cflmin* and *cflmax* are non-negative, the usual one-dimensional (inviscid) CFL number definition using the maximum eigenvalue is used.

The default values are both 0.0.

Examples

Set the CFL number to 1.5 in all zones.

```
CFL# CFL 1.5
```

Set the CFL number to 1.5 in zones 1 and 5, and 0.7 in zones 2, 3, and 4.

```
CFL# CFL 1.5 ZONE 1,5
CFL# CFL 0.7 ZONE 2:4
```

The following example sets the CFL number to 1.0, computes the corresponding time step at every point in the grid, finds the minimum of those time steps, and resets the time step at every point to that minimum value. This allows the time step to be set for an unsteady flow problem by specifying a CFL number. Note that even though this may only be done for single-zone grids, the syntax requires that the zone be explicitly specified.

```
CFL# CFL 1.0 ZONE 1 1
```

Set the time step equal to 1×10^{-6} seconds at every point.

```
CFL# SECONDS 0.000001
```

The next example sets the CFL number to 0.5 for the first 500 iterations, then increases it by a factor of 1.1 every 100 iterations. The maximum value allowed will be 2.0, so the actual final CFL number will be some value just below 2.0. (For this example, it's 1.899.)

```
CFL# CFL INCREMENT 0.5 2.0 1.1 100 501
```

This example uses the mode 3 time step calculation procedure to set the time step parameter to 0.1, and defaults the remaining input parameters, causing the actual time step to be scaled by the local metric Jacobian with the 0.005 fudge factor, with no minimum or maximum value.

```
CFL# MODE 3 DT 0.1
```

In this example the mode 3 time step calculation procedure is used to set the time step parameter to 0.5. The actual time step will be scaled by the local metric Jacobian with the 0.005 fudge factor, with minimum and maximum CFL numbers of 1.0 and 5.0. Since the minimum and maximum values are specified as positive numbers, the standard one-dimensional (inviscid) CFL number definition using the maximum eigenvalue is used.

```
CFL# MODE 3 DT 0.5 CFLMIN 1.0 CFLMAX 5.0
```

The next example uses the mode 3 time step calculation procedure to set the time step parameter to 0.05. The actual time step will be scaled by the local metric Jacobian without the 0.005 fudge factor, with minimum and maximum CFL numbers of 1.0 and 5.0. Since the minimum and maximum values are specified as negative numbers, the CFL definition of Gnoffo is used, with a viscous correction by Tannehill.

```
CFL# MODE 3 DT 0.05 ITIME 2 CFLMIN -1.0 CFLMAX -5.0
```

Use the mode 3 time step calculation procedure to set the CFL number to a constant value of 1.3.

```
CFL# MODE 3 ITIME 3 CFLMAX 1.3
```

Use the mode 3 time step calculation procedure for an unsteady case, with the nondimensional time step set to a constant value of 0.05.

```
CFL# MODE 3 ITIME 0 DT 0.05
```

The following example uses the mode 3 time step calculation procedure for an unsteady case, again with a specified nondimensional time step of 0.05. In this case, however, if the specified time step results in a CFL number at some point in the flow field less than 0.25 or greater than 5.0, the actual time step throughout the flow field will be reset to the corresponding value.

```
CFL# MODE 3 ITIME 0 DT 0.05 CFLMIN 0.25 CFLMAX 5.0
```

See Also: CROSSFLOW

CGNSBASE — Use CGNS files

`CGNSBASE` *basename*

This keyword specifies that the grid and flow data are obtained from, and written to, CGNS (CFD General Notation System) files, not common files. The user-specified *basename* is the name of the `CGNSBase_t` node in the grid and flow files.[22]

It should be noted that, strictly speaking, there is no notion of a CGNS *file*, only of a CGNS *database* implemented within one or more ADF (Advanced Data Format) files. We nevertheless use the terminology "CGNS file" in the WIND documentation. In addition, a true CGNS database has a single `CGNSBase_t` node as its root. In the current WIND implementation, however, the grid and flow solution are stored in separate files, with both files having their own `CGNSBase_t` node with the same *basename*.

If the `CGNSBASE` keyword is not used, the grid and flow data are assumed to be in common files.

WIND supports the most common CGNS features, but not all. Some specific limitations on the implementation of CGNS in the WIND code are listed below.

- Many WIND input options may be specified on a zonal basis, using the zone number to identify the zone. In the CGNS standard, zones are identified by a name. When CGNS files are used in WIND, it is assumed that the zone number maps to the zone's position in an alphanumerically sorted list of zone names.

  Note that the naming convention `Zone`*n*, where *n* is the zone number, is alphanumeric only up to `Zone9`. `Zone10` through `Zone19` would get sorted between `Zone1` and `Zone2`, and so on. However, spaces are allowed in names, so "`Zone` *n*", with two spaces, (e.g., `Zone  1`, `Zone  2`, ..., `Zone 99`, `Zone100`, ...) is alphanumeric up to `Zone999`.

- The CGNS default names "`GridCoordinates`" and "`FlowSolution`" are used for the CGNS `GridCoordinates_t` and `FlowSolution_t` nodes. For each zone, only one grid and one flow solution are stored.

- In a CGNS database, the scaling data must be stored with the data itself, but the units to be used may be stored with the data or higher in the node tree. The WIND implementation assumes that the units are stored with the data.

- If grid velocities are stored in the grid file, it is assumed that the units are ft/sec.

- The CGNS nodes for storing time-dependent data are not yet supported in WIND.

- For overlapping grids, CGNS allows hole points to be identified using multiple `OversetHoles_t` nodes. WIND currently assumes that all hole points in a zone are identified in a single `OversetHoles_t` node, using a `PointList`.

- WIND writes convergence and time history information to the list output (*.lis*) and time history (*.cth*) files, respectively, not to a CGNS file.

There are also some WIND features and capabilities that are not supported by the current CGNS standard. These include chemistry and MFD flows, specified boundary layer transition, and bleed boundary conditions. Several proposals are pending to extend the CGNS standard, but for now CGNS files created by WIND for these types of flows may not be fully CGNS compliant, and should be considered unique to the WIND implementation.

[22] Detailed information on the CGNS standard may be found at the CGNS web site, at http://cgns.sourceforge.net/.

CHEMISTRY — Chemistry model selection (block)

```
CHEMISTRY
   FILE {filename [LOCAL]}
   {FROZEN | FINITE RATE}
   [FUEL AIR RATIO value]
   [LIU-VINOKUR]
   MASS [FRACTIONS] {AIR | sp1 sp2 ...spn}
   OMIT [THIRD-BODY] S1 S2 ...Sm ZONE range1[,range2[, ... rangen]]
   [VISCOSITY {SUTHERLAND | WILKE | KEYE | CONSTANT}]
ENDCHEMISTRY
```

This option allows the user to specify the real gas chemistry mode and input data for the desired species and reactions.

Note: The ARBITRARY INFLOW keyword block, if used, must come after the CHEMISTRY keyword block in the input data (*.dat*) file.

The various elements of the CHEMISTRY input block are defined as follows:

`CHEMISTRY`

Defines the beginning of the CHEMISTRY block

`FILE {`*filename* `[LOCAL]}`

Specifies the chemistry data file containing the thermodynamic data, finite rate coefficients, and transport property data for Wilkes' law. (See Section 7.10 for the file format.) If the chemistry data file is located in the directory that WIND files (*.cfl*, *.cgd*, etc.) are running the keyword LOCAL must be specified.

`{FROZEN | FINITE RATE}`

FROZEN	Selects frozen chemistry
FINITE RATE	Selects finite rate chemistry

`FUEL AIR RATIO` *value*

value specifies the fuel-air ratio for H_2-air combustion (two-reaction global model).

`LIU-VINOKUR`

Selects the Liu and Vinokur equilibrium air curve fits. Valid to 50000 K, does not output species.

`MASS [FRACTIONS] {AIR |` sp_1 sp_2 `...`sp_n`}`

Specifies the freestream (reference) mass fractions of the species. The keyword `AIR` indicates use of air mass fractions.

`OMIT [THIRD-BODY]` S_1 S_2 ...S_m `ZONE` *range1*`[,`*range2*`[, ... rangen]]`

This keyword allows control of the reacting species in a finite-rate chemistry solution on a zonal basis. Reactions leading to the production or destruction of the species given by S_1 through S_m will be eliminated in the selected zones. In addition, if the `THIRD-BODY` keyword parameter is used, *any* reaction involving the named species will be eliminated. The species will still be convected and diffused, maintaining conservation at zone boundaries.

Eliminating reactions in zones where they aren't relevant improves the overall efficiency of the calculation in two ways. First, the time required to perform the unnecessary computations is eliminated. And second, the stiffness of the system of equations is reduced, leading to faster convergence and greater stability.

In the zone specification, the range parameter(s) may be

zonenum	Selects zone *zonenum*
begzone`:`*endzone*	Selects all zones from *begzone* to *endzone*
`ALL`	Selects all zones

Multiple `OMIT` keywords are allowed, with conflicting input resolved by using the latter of the conflicting entries.

`VISCOSITY {SUTHERLAND | WILKE | KEYE | CONSTANT}`

Selects the transport property equations. See the `VISCOSITY` keyword for details.

`ENDCHEMISTRY`

Defines the end of the `CHEMISTRY` block.

See Also: `TEST 71`

General and Standard Chemistry Packages

The chemistry data files supplied with the WIND code are summarized in Table 1.[23]

Table 1: General and Standard Chemistry Packages

Species	File(s)	T_{max}	*Nc*	*Nr*	*3d*
O_2, NO, O, N, N_2	*air-5sp-std-06k.chm* *air-5sp-gen-06k.chm*	6000 K	1	5	V
O_2, NO, O, N, N_2	*air-5sp-std-15k.chm* *air-5sp-gen-15k.chm*	15000 K	3	5	V
O_2, NO, O, N, N_2	*air-5sp-std-30k.chm* *air-5sp-gen-30k.chm*	30000 K	5	5	V
O_2, NO, O, N, NO^+, e^-, N_2	*air-7sp-std-06k.chm*	6000 K	1	6	V
O_2, NO, O, N, NO^+, e^-, N_2	*air-7sp-std-15k.chm* *air-7sp-gen-15k.chm*	15000 K	3	6	V
O_2, NO, O, N, NO^+, e^-, N_2	*air-7sp-std-30k.chm* *air-7sp-gen-30k.chm*	30000 K	5	6	V
O_2, OH, H_2, H_2O, N_2	*h2air-5sp-std-06k.chm* *h2air-5sp-std-15k.chm*	6000 K 15000 K	1 3	2	
O_2, H, H_2, H_2O, N_2	Not yet available				
O_2, H, H_2, H_2O, OH, O, N_2	*h2air-7sp-std-15k.chm* *h2air-7sp-gen-15k.chm* *h2air-7sp-bak-15k.chm*	15000 K	3	8	
O_2, CO_2, H_2O, NO_2, N_2	Not yet available				

[23] In Table 1, the symbol e^- is electron density, and NO^+ is ionized nitrogen oxide. Also, T_{max} is the maximum temperature at which the curve fits are valid, *Nc* is the number of thermodynamic curve fit segments, *Nr* is the number of reaction equations, and *3d* denotes Variable or Average 3rd-body efficiency.

/ — Comment lines

In the extended portion of the file, a "comment line" may be placed anywhere. This line is denoted by a / in column 1. The remainder of the line is disregarded by the input routines, as if the comment line did not exist. The purpose of this option is to allow the user to place explanatory notes within the flow condition data file. A blank line is also ignored.

COMPRESSOR FACE — Outflow boundaries, compressor face

```
COMPRESSOR FACE {CHUNG mach zone | BOEING mode val1 [val2] zone | \
                 PAYNTER mach val1 val2 zone | SAJBEN mach angh angc zone} \
                [ZONE] range1[,range2[, ...  rangen]]
```

This keyword allows an outflow boundary to be modeled as a compressor face. In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
ALL	Selects all zones

There are several models available, as described below. All are based on the observation that turbine engine conditions set the corrected mass flow, and that this corresponds directly to the average Mach number at the compressor face. These boundary conditions have been implemented mainly for the analysis of unsteady flow; however, they have also been shown to be robust for the establishment of steady-state, supercritical inlet flows.

```
COMPRESSOR FACE CHUNG mach [ZONE] range1[,range2[, ...  rangen]]
```

The CHUNG keyword option uses the Chung-Cole model (Chung and Cole, 1996). Based on the specified mass-averaged Mach number *mach*, and using the computed local Mach number and total pressure, a new local static pressure is computed.

Note: This boundary condition is identical in its effect on the flow to the DOWNSTREAM MACH keyword.

```
COMPRESSOR FACE BOEING mode val1 [val2] [ZONE] range1[,range2[, ...  rangen]]
```

The BOEING keyword option uses uses a model developed by Mayer and Paynter (1994). The parameters to be specified as *val1* and *val2* depend on *mode*, as shown in the following table.

mode	*val1*	*val2*
1	Average Mach number	—
2	Corrected mass flux per unit area, $lb_m/sec/ft^2$	—
3	Nominal Mach number	Nominal total temperature, °R

Modes 1 and 2 are equivalent, and are useful in obtaining initial flow fields, and for simulating instantaneous changes at the compressor face via a restart. Mode 3 is used to set a constant volumetric flow condition. Typically, an initial condition is obtained using mode 1 and becomes the nominal condition. Then *val1* is set to the Mach number used in mode 1 and *val2* is set to the freestream total temperature, i.e., it is assumed that the total temperature at the compressor face is equal to the freestream value. If the average total temperature at the compressor face does not change, then modes 1 and 3 will yield identical results. However, if the average total temperature at the compressor face changes due to a freestream disturbance, for example, then the corrected flow rate will be adjusted to maintain a constant volumetric flow rate.

`COMPRESSOR FACE PAYNTER` *mach val1 val2* `[ZONE]` *range1*`[,`*range2*`[, ... rangen]]`

The Paynter compressor face model (Paynter, 1998) computes unsteady acoustic reflections from a compressor face subjected to acoustic and convective disturbances. This model is intended to be used for time-accurate simulations of unsteady inlet flow. The CFL number should be less than one. The flow is assumed to be subsonic and axial as it enters the compressor face.

The model assumes the compressor face is positioned within an annular duct with a hub and case, and consists of a single row of axial flow compressor blades. The conditions at the compressor face are determined by the convective velocity response coefficient α, and the acoustic response coefficient β. The definitions of α and β depend on the passage Mach number $M_{passage}$, which is given by

$$M_{passage} = M_{cf} / \cos\Gamma$$

where M_{cf} is the compressor face Mach number, and Γ is the local stagger angle.

For subsonic flow ($M_{passage} < 1$),

$$\alpha = \frac{\gamma M_{cf}^2}{1 - M_{cf}} \tan\left(\frac{\Gamma}{2}\right) \tan\Gamma$$

$$\beta = \tan^2\left(\frac{\Gamma}{2}\right)\left(\frac{1 + M_{cf}}{1 - M_{cf}}\right)$$

where γ is the specific heat ratio. For supersonic flow ($M_{passage} > 1$),

$$\alpha = \gamma M_{cf}$$

$$\beta = 1$$

Details on the implementation of this boundary condition are given by Slater and Paynter (2000).

In the WIND code, α and β may be treated as constant and specified directly, or computed using the above equations. The local stagger angle Γ is determined by assuming a linear variation between specified values of the stagger angles at the hub and case. The solidity of the blade row is assumed to be greater than one.

The absolute value of the keyword parameter *mach* sets the nominal or average compressor face Mach number M_{cf}. The meanings of *val1* and *val2* depend on the sign of *mach*:

mach	*val1*	*val2*
< 0	Stagger angle (degrees) of the compressor fan blade at the hub	Stagger angle (degrees) of the compressor fan blade at the case
> 0	Constant convective velocity response coefficient, α	Constant convective velocity response coefficient, β

`COMPRESSOR FACE SAJBEN` *mach angh angc* `[ZONE]` *range1*`[,`*range2*`[, ... rangen]]`

The Sajben compressor face model is basically the same as the Paynter model, except that Sajben's expression for the acoustic response coefficient is used for subsonic flow (Sajben, 1999).

Sajben's expression for the acoustic response coefficient may be written as

$$\beta = \frac{M_{cf} \tan^2\Gamma}{2(1 + M_{cf}) + M_{cf} \tan^2\Gamma}$$

where M_{cf} is the compressor face Mach number, and Γ is the local stagger angle.

With the `COMPRESSOR FACE SAJBEN` keyword, the keyword parameter *mach* is the nominal or average compressor face Mach number M_{cf}, and *angh* and *angc* are the stagger angles in degrees of the compressor fan blade at the hub amd case.

Examples

```
COMPRESSOR FACE CHUNG 0.3 1
COMPRESSOR FACE BOEING 1 0.3 1
COMPRESSOR FACE BOEING 2 0.5516 1
COMPRESSOR FACE BOEING 3 0.3 520.0 1
```

See Also: DOWNSTREAM MACH, DOWNSTREAM PRESSURE, MASS FLOW

CONVERGE — Controls convergence

```
CONVERGE {LEVEL | ORDER} val
```

This keyword allows the user to specify the convergence criteria used for a run. The two possible options are defined below.

`LEVEL` Check for maximum residual less than *val*.

`ORDER` Check for reducing maximum residual by *val* orders of magnitude.

The default is a four-order of magnitude decrease in the maximum residual.

If Newton iteration is being used, this keyword specifies the convergence criteria within a Newton time step. The overall global convergence criteria is specified using the `NEWTON` keyword.

See Also: `TEST 128`, `NEWTON`

COUPLING — Zone coupling mode specification

```
COUPLING [MODE] {CHARACTERISTIC | ROE [HIGH | LOW]}
```

This keyword controls the zone coupling algorithm used for multi-zone solutions. The two possible options are defined below.

`CHARACTERISTIC` This is NASTD's original zone coupling algorithm that uses one-dimensional characteristic flow theory to set boundary flowfield variables based on local flow direction and strength. These boundary variables are then transferred between zones using interpolation factors stored in the grid file (the *.cgd* file).

`ROE` This newer zone coupling algorithm is more consistent with Roe's flux-difference splitting scheme, which is WIND's default explicit operator. Instead of transferring flowfield variables between zones, this algorithm transfers flux cell "interface states," which are critical quantities in Roe's scheme. In this coupling mode, the zone boundary may be thought of as a "cell interface" in Roe's scheme.

The `HIGH` option, which is the default, turns on higher-order coupling at zone boundaries. With higher-order coupling the solution derivatives are also passed between coupled zones. This results in a slight increase in memory requirements. The derivative information is used to increase the accuracy of the coupling scheme and to improve robustness at coupled boundaries through the application of a TVD operator.

Roe coupling (`HIGH` or `LOW`) requires that the Roe, Van Leer, or HLLE explicit operator be used. If `RHS CENTRAL` or `RHS COAKLEY` is specified, characteristic zone coupling will automatically be used.

In addition, since a TVD operator is used with higher-order Roe coupling, the default `HIGH` option cannot be used with a third-order fully upwind explicit operator, or with any of the fourth- or fifth-order explicit operators.

See Also: `RHS`, `TVD`

CROSSFLOW — Crossflow CFL factor

```
CROSSFLOW [CFL] [FACTOR] val [zone_selector]
```

The `CROSSFLOW` keyword sets the cross-flow CFL factor to *val* in the specified zones.

For most WIND calculations, the time step is based on a constant CFL number throughout the flow field. Thus, for non-uniform grids, the solution advances at different rates in different parts of the grid. Because the CFL number is directly proportional to the physical time step and inversely proportional to the local grid spacing, the physical time step is small where the grid spacing is small, in order to maintain a constant CFL number. For example, in boundary layers and shear layers, where the grid is often closely packed for better resolution, the solution advances more slowly than in other parts of the flowfield. In fact, most of the iterations in a viscous solution are spent converging viscous regions.

To speed up convergence in these regions, each coordinate direction's eigenvalues are multipled by a factor before determining the time step associated with the local spacing and flow conditions. This factor varies from 1.0 when the flow is along the coordinate direction, to the specified value when the flow is perpendicular to the coordinate direction. The far-field CFL number is generally the value specified in the input data file with the `CFL#` keyword, but near the wall the code effectively increases the time step in the boundary layer by the specified factor. This should increase convergence of the boundary layer.

The default cross-flow CFL factor is 2.0 for three-dimensional flow, and 1.0 (i.e., no time step increase in the boundary layer) for two-dimensional flow and axisymmetric flow.

See Also: `CFL#`

CYCLES — Number of solution cycles

```
CYCLES ncyc [PRINT FREQUENCY freq]
```

This keyword is used to set the number of cycles *ncyc*, and is required in the input data file. By default each cycle has five flowfield iterations in all zones, but this may be changed using the ITERATIONS keyword.

Cycle time information will be written to the *.lis* file every *freq* cycles, starting with the first cycle. The default is every cycle. Residuals will also be written to the *.lis* file every *freq* cycles, starting with the first cycle, but only for iterations consistent with the PRINT FREQUENCY specified with the ITERATIONS keyword.

Note that this does not affect integrated properties specified using the LOADS keyword block. The print frequency for integrated properties is controlled by the PRINT keyword in the LOADS keyword block.

Example

If the user specifies

```
CYCLES 1000 PRINT FREQUENCY 10
ITERATIONS PER CYCLE 10 PRINT FREQUENCY 10
```

cycle time information will be printed for cycles 1, 11, 21, etc. Residuals will be printed for iterations 10 (in cycle 1), 110 (cycle 11), 210 (cycle 21), etc.

See Also: ITERATIONS, LOADS

DOWNSTREAM MACH — Outflow boundaries, Mach number

```
DOWNSTREAM MACH [NUMBER] mach [ZONE] range1[,range2[, ... rangen]]
```

This keyword allows the specification of a mass-averaged subsonic Mach number *mach* at outflow boundaries. Based on the specified mass-averaged Mach number, and using the computed local Mach number and total pressure, a new local static pressure is computed. This boundary condition simulates the uniform Mach number characteristics observed at the compressor faces of turbine engines, and is identical in its effect on the flow to the `COMPRESSOR FACE CHUNG` keyword.

In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
`ALL`	Selects all zones

See Also: `COMPRESSOR FACE`, `DOWNSTREAM PRESSURE`, `MASS FLOW`

DOWNSTREAM PRESSURE — Outflow boundaries, pressure

```
DOWNSTREAM PRESSURE { \
   value [EXTRAPOLATE [SUPERSONIC]] | \
   FREESTREAM [EXTRAPOLATE [SUPERSONIC]] | \
   EXTRAPOLATE [ALWAYS] | \
   value VARIABLE i j k [RELAXER rlxr] | \
   value UNSTEADY {SINUSOIDAL | USERSPEC} Δp freq phase \
   } [ORDER {ZERO|0|ONE|1}] [ZONE] range1[,range2[, ...  rangen]]
```

This keyword controls the specification of a pressure boundary condition at outflow boundaries. In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
ALL	Selects all zones

There are five possible modes, defined as follows:

```
DOWNSTREAM PRESSURE value [EXTRAPOLATE [SUPERSONIC]] \
   [ORDER {ZERO|0|ONE|1}] [ZONE] range1[,range2[, ...  rangen]]
```

This mode specifies the pressure *value* (in psi) to be applied at the outflow boundary in the specified zone(s). The `EXTRAPOLATE [SUPERSONIC]` option tells WIND to extrapolate all flow conditions to the boundary for points where the flow is supersonic. Either zeroth- or first-order extrapolation will be used, as specified by `ORDER`.

For boundary points where the flow is subsonic,

1. The pressure at the boundary is set to the user-specified value.
2. For each boundary point, the density and momentum, plus the effective gamma, compressibility factor, and speed of sound, are extrapolated from the interior to the boundary using either zeroth- or first-order extrapolation, as specified.
3. The energy at each boundary point is computed, consistent with the user-specified pressure and the extrapolated values of density, etc.

```
DOWNSTREAM PRESSURE FREESTREAM [EXTRAPOLATE [SUPERSONIC]] \
   [ORDER {ZERO|0|ONE|1}] [ZONE] range1[,range2[, ...  rangen]]
```

This mode is the same as the previous one, except that the pressure value is taken from the FREESTREAM keyword.

```
DOWNSTREAM PRESSURE EXTRAPOLATE [ALWAYS] \
   [ORDER {ZERO|0|ONE|1}] [ZONE] range1[,range2[, ...  rangen]]
```

This mode extrapolates all flow conditions for all points on the boundary, regardless of whether or not the flow there is supersonic.

```
DOWNSTREAM PRESSURE value VARIABLE i j k [RELAXER rlxr] \
   [ORDER {ZERO|0|ONE|1}] [ZONE] range1[,range2[, ...  rangen]]
```

This mode imposes the pressure *value* (in psi) only at the specified (*i,j,k*) boundary point in the specified zone(s). The indices (*i,j,k*) should correspond to a point on the outflow boundary, although at present no check is made to verify that they do. The pressure at the remaining boundary points will spatially vary according to the spatial variation of pressure at the solution points adjacent to the boundary, and may be over- or under-relaxed using the input relaxation factor *rlxr*. The default value for *rlxr* is 1.0 (i.e., no relaxation).

The procedure is as follows:

1. A "reference" pressure p_{ref} at the specified (i, j, k) boundary point is found by extrapolating the interior pressure values to the boundary, using either zeroth- or first-order extrapolation, as specified.
2. For each boundary point, the density and momentum, plus the pressure, effective gamma, compressibility factor, and speed of sound, are extrapolated from the interior to the boundary using either zeroth- or first-order extrapolation, as specified.
3. At each boundary point, the pressure is re-computed using

   $$p = p_{ext}\frac{p_{nom}}{p_{ref}}$$

   where p_{ext} is the extrapolated value from step 2, p_{nom} is the nominal value specified with the `DOWNSTREAM PRESSURE` keyword, and p_{ref} is the "reference" value from step 1.
4. The actual imposed pressure at each point is relaxed using

   $$p = rp + (1 - r)p_{old}$$

   where r is the input relaxation factor *rlxr*, and p_{old} is the pressure value from the previous iteration.
5. The energy at each boundary point is computed, consistent with the pressure value from step 4, and the extrapolated values of density, etc., from step 2.

```
DOWNSTREAM PRESSURE value UNSTEADY \
   {SINUSOIDAL | USERSPEC} Δp freq phase \
   [ORDER {ZERO|0|ONE|1}] [ZONE] range1[,range2[, ...  rangen]]
```

This mode allows an unsteady pressure to be applied at the outflow boundary in the specified zone(s). A sinusoidal or user-defined oscillation may be specified. The parameters are defined as:

value Baseline pressure, p_s (psi)

Δp	Amplitude of oscillation (psi)
freq	Frequency of oscillation, ω (Hz)
phase	Phase angle of oscillation, ϕ (deg)

If `SINUSOIDAL` is used, specifying a sinusoidal pressure oscillation, the pressure is computed from

$$p = p_s + \Delta p \, \sin[2\pi(\omega t + \phi)]$$

where p is the pressure and t is the time.

If `USERSPEC` is used, specifying a user-defined pressure oscillation, a table must be input immediately following the `DOWNSTREAM PRESSURE` keyword, with the following format:

```
PROFILE npts
per(1)     amp(1)
per(2)     amp(2)
...
per(npts)  amp(npts)
```

where *npts* is the number of points in the profile (max 65), *per* is the normalized period (i.e., from 0.0 to 1.0), and *amp* is the normalized amplitude (i.e., from −1.0 to 1.0). As an example, the following normalized pressure oscillation:

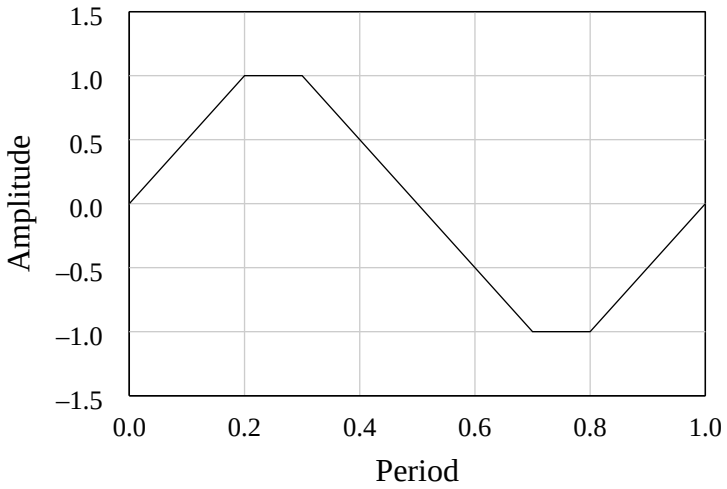


Figure 7: Example user-specified unsteady pressure oscillation

could be specified using the following profile input:

```
PROFILE 6
0.0   0.0
0.2   1.0
0.3   1.0
0.7  -1.0
0.8  -1.0
1.0   0.0
```

Once the downstream pressure is determined for the current time level, the density and momentum at each boundary point, plus the effective gamma, compressibility factor, and speed of sound, are extrapolated from the interior to the boundary using either zeroth- or first-order extrapolation, as specified. The energy at each boundary point is then computed, consistent with the pressure and the extrapolated values of density, etc.

Extrapolation Notes

The default for all extrapolation is zeroth-order (i.e., conditions at the boundary are set to the values at the computational plane adjacent to the boundary). This results in a discontinuous slope in flow values near the outflow boundary, which may be important for flows with significant streamwise pressure gradients. First-order extrapolation yields smoother results.

For flows with little or no streamwise pressure gradient near the outflow boundary, the results using zeroth- and first-order extrapolation are essentially identical. Convergence rates and the final residual values are generally better with zeroth-order extrapolation, however, so the default zeroth-order extrapolation is recommended.

For flows with significant streamwise pressure gradients near outflow boundaries, zeroth-order extrapolation can give poor results at the outflow boundary, and in some cases these can affect values at the inflow boundary. First-order extrapolation is thus recommended for these flows.

See Also: COMPRESSOR FACE, DOWNSTREAM MACH, MASS FLOW, FREESTREAM

DQ — ΔQ limiter

```
DQ [LIMITER] [OFF | ON [DRMAX drmax] [DTMAX dtmax]]
```

This keyword may be used to limit the change in density and/or temperature over a single iteration. This may be helpful, especially in the initial stages of a calculation, in maintaining stability, and allow the use of larger CFL numbers than might otherwise be possible.

`DRMAX` *drmax* — *drmax* specifies the fractional change allowed in the density over an iteration. I.e., specifying "`DQ LIMITER ON DRMAX 0.2`" means the density will not be permitted to change by more than ±20% from one iteration to the next.

`DTMAX` *dtmax* — *dtmax* specifies the fractional change allowed in the temperature over an iteration.

The default is to not limit the change in density or temperature. When limiting is turned on, the defaults for *drmax* and *dtmax* are both 0.5. Thus, if just "`DQ LIMITER ON`" is specified, both density and temperature will be allowed to change by no more than ±50% over an iteration.

The use of the `DQ LIMITER` keyword becomes somewhat problematic when multiple sub-iterations are used for the Navier-Stokes equations (using the `NAVIER-STOKES ITERATIONS` keyword) and/or the turbulence model equations (using the `ITERATIONS` option with the `TURBULENCE` keyword). By default, the number of sub-iterations for both the Navier-Stokes and turbulence model equations is one. In this case, each iteration for the Navier-Stokes equations is followed by an iteration for the turbulence model equations, and if the limiter is applied to the Navier-Stokes solution, it is also applied to the turbulence model solution.

If the number of Navier-Stokes sub-iterations is one, and if multiple sub-iterations are used for the turbulence model equations, then if the limiter is applied to the Navier-Stokes solution, it is applied to each turbulence model sub-iteration.

Also, if multiple Navier-Stokes sub-iterations are used, it's possible that the computed flow field is such that the limiter is applied during some sub-iterations, but not others. In this case, if the limiter is applied during the final Navier-Stokes sub-iteration, it is again applied to each turbulence model sub-iteration. If the limiter is not applied during the final Navier-Stokes sub-iteration, it will not be applied at all during the turbulence model iteration.

For these reasons, it is recommended that the number of sub-iterations for the Navier-Stokes and turbulence model equations be left at their default values of one when the `DQ LIMITER` keyword is used.

END — Termination

```
[END]
```

This keyword, which must begin in column 1, may be used to end the input data file.

FIXER — Instability smoothing

`FIXER [PRINT] [`*zone_selector*`]`

WIND's `FIXER` capability searches for computational points at which there are negative values of density or internal energy (due to numerical instabilities) and replaces the faulty values with averages from nearby j and k points in the same i-plane. (I.e. the smoothing is two-dimensional; values from adjacent i-planes are not used.) The process is iterative, with multiple smoothing passes. Only "active" points are examined, not wall, coupled boundary, hole, or fringe points.

In a given i-plane, if more than 50% of the points are "bad", or if the smoothing process fails to remove all the negative values, the run is aborted.

Using `FIXER` may help the solution through difficult transients, but *it may also simply smear out instabilities over a larger portion of the flowfield.* You should maintain a careful watch over the solution.

For each i-plane containing negative values of density or internal energy, the zone, iteration number, i index, and number of bad points will be written to the list output (*.lis*) file. If the `PRINT` option is specified, the listing will also include the j and k indices of each bad point.

FREESTREAM — Freestream conditions

```
FREESTREAM {STATIC | TOTAL} [CONDITIONS] M P T α β
```

This keyword allows input of freestream flow conditions. The user-specified parameters are defined as:

M	Mach number
P	Static or total pressure (psi), depending on the `STATIC | TOTAL` keyword
T	Static or total temperature (°R), depending on the `STATIC | TOTAL` keyword
α, β	Angles of attack and yaw (degrees)

Note that, if U is the freestream velocity magnitude, α and β are related to the Cartesian velocity components by:

$$w = U \sin\beta$$
$$u = U \cos\beta \cos\alpha$$
$$v = U \cos\beta \sin\alpha$$

Note: The local flow angles (*localpha* and *locbeta* in the CFPOST post-processing package) are defined relative to the x, y, z coordinates and are given by:

$$localpha = \tan^{-1}(v/u)$$
$$locbeta = \tan^{-1}(w/u)$$

Thus if β is not equal to 0, *locbeta* is not equal to β.

Note: WIND assumes α is rotation about the z axis. That is, y is up.

FRINGE — Solution mode at fringe points

```
FRINGE [MODE] FROZEN [zone_selector]
```

For overlapping zonal boundaries, this keyword may be used to freeze the solution at fringe points during a cycle. Solution values at these points are then updated at the end of the cycle.

`FRINGE FROZEN` *must* be used for double-fringe boundaries. Running a double-fringe grid without this keyword may lead to erroneous results.

`FRINGE FROZEN` may also be specified for single-fringe boundaries, but is not required. The default for single-fringe boundaries is to update the solution at fringe points every iteration.

Using double fringes increases the solution accuracy at overlapping boundaries. Note, though, that by freezing the solution over a cycle, the solution at fringe points will lag in time behind the solution at interior points. This could potentially lead to instabilities during transients, or a reduction in the convergence rate if a large number of iterations per cycle are specified. However, testing on a limited number of cases has not revealed any problems when using up to five iterations per cycle.

Double-fringe boundaries may be created with GMAN, by first selecting double-fringe mode using (in GMAN's graphical mode) `MODIFY BNDY > GENERATE FRINGE > SET FRINGE-MOD > DOUBLE FRINGE`, then generating the fringe.

GAS — Gas property specification

```
GAS gam prl prt gasc
```

This option allows the user to input different values of γ, laminar and turbulent Prandtl numbers, and gas constant. The parameters are defined as:

gam	Ratio of specific heats $\gamma = c_p/c_v$ (default is 1.4)
prl	Laminar Prandtl number (default is 0.72)
prt	Turbulent Prandtl number (default is 0.90)
gasc	Gas constant R (default is 1716 ft^2/sec^2-°R)

Note: The ARBITRARY INFLOW keyword block, if used, must come after the GAS keyword in the input data (*.dat*) file.

GRAVITY — Add gravity body forces

`GRAVITY` x_c y_c z_c g_x g_y g_z

The `GRAVITY` keyword allows the user to include the effects of gravity on the fluid.

x_c, y_c, z_c Coordinates of a point on the "ground" plane (normal to the gravity vector) used to define the location of zero potential energy, in grid units

g_x, g_y, g_z Components of the gravity vector, in ft/sec^2. For example, if the y direction is normal to the earth's surface, these values might be 0.0, −32.2, 0.0.

GRID LIMITER — Grid limiting capability

```
GRID LIMITER {ANGLE angle | OFF} [zone_selector]
```

WIND's grid limiting capability switches the explicit operator to a first-order scheme in the presence of grid turning greater than a specified amount. This capability may be needed to reduce numerical instabilities near wing tips, leading edges, and other geometric features which may require extreme amounts of grid turning.

The parameter *angle* is the number of degrees off of a straight line allowed before limiting the scheme to first order.

See Also: TEST 110, TVD

HISTORY — Time history flowfield variable tracking

```
HISTORY
   VARIABLE [M] [p] [T] [u] [v] [w] [radius] [DeltaP] [x] [y] [z] \
            [rho] [rho*u] [rho*v] ...
   [FREQUENCY f]
   REGION nzn ibeg iend jbeg jend kbeg kend
   REGION nzn NODE nodenum
   REGION nzn CELL cellnum
   ...
ENDHISTORY ...
```

For unsteady flow problems, the `HISTORY` keyword block allows the user to track the temporal evolution of variables at discrete specified points. The specified flow quantities are written to a common time history (*.cth*) file every f iterations. Upon completion of a time history run, the auxiliary program *thplt* may be used to analyze the data and/or to create a GENPLOT file that may then be plotted using CFPOST.[24] The *.cth* file contains reference and scaling data. Thus, when *thplt* is used to create a GENPLOT file, the time history variables will be dimensional.

For the results to be strictly valid, all zones must run with the same time step and with one iteration per cycle. However, in recognition that this is not always feasible, this capability can be used with only the stipulation that the zone(s) in which temporal evolution information is desired be run with a constant time step.

The *.cth* file itself does not support direct I/O. Therefore, when WIND is being run in parallel mode, each individual processor maintains the requested time history for its zones, but calls the master process to update the *.cth* file whenever the flow (*.cfl*) file is updated, and whenever the time history memory buffer is filled. This forces the *.cth* file to be consistent with the *.cfl* file.

When restarting a time-accurate job with the same input conditions, the existing *.cth* file will normally be extended. However, any changes to the time history points or variables, or the flow reference state, will require starting with a fresh *.cth* file. In this case, the name of the previous *.cth* file should be changed. If time history or reference condition changes have been made, and an older *.cth* file exists with the same base name, the WIND run will abort. In any case, *thplt*, the time history post-processing tool, can read multiple *.cth* files to combine the different time intervals.

The various elements of the `HISTORY` keyword block are defined as follows:

```
HISTORY
```

Defines the beginning of the `HISTORY` block

```
VARIABLE [M] [p] [T] [u] [v] [w] [radius] [DeltaP] [x] [y] [z] \
         [rho] [rho*u] [rho*v] ...
```

Specifies the variables to be recorded. At least one must be specified, and the variable names are *case-sensitive*. The first eight variables shown above are defined as follows:

[24] Because of changes to the format of the *.cth* file, the *timplt* utility used to analyze *.cth* files created by WIND 5.51 and earlier will not work with *.cth* files created by the current WIND code.

`M`	Mach number
`p`	Static pressure
`T`	Static temperature
`u`	Velocity in the x direction
`v`	Velocity in the y direction
`w`	Velocity in the z direction
`radius`	Distance from a specified reference point to the location of a shock front. See the section Front Tracking Mode for details.
`DeltaP`	Monotone pressure rise across the shock front. See the section Front Tracking Mode for details.

In addition to the variables listed above, any variable in the `q` array may be specified. These of course depend on the equations being solved. The possible `q` variables are listed in Table 2, in groups, along with an indication of when they are available.

Table 2: `HISTORY` Keyword — `q` Variables

Name	Definition	When Used
—*Navier-Stokes equations*		
`rho`	Static density	Always
`rho*u`	Momentum in the x direction	
`rho*v`	Momentum in the y direction	
`rho*w`	Momentum in the z direction	
`rho*e0`	Total energy	
—*Auxiliary parameters*		
`beta`	Effective specific heat ratio	Always
`Z`	Compressibility factor	
`a`	Speed of sound	
`kappa`	Thermal conductivity	
`dt`	Time step	
—*Viscosity*		
`mul`	Laminar viscosity coefficient	Viscous flows
`mut`	Turbulent viscosity coefficient	
—*Turbulence equations*		
`anut`	Eddy viscosity	Baldwin-Barth and Spalart-Allmaras turbulence models
`k`	Turbulent kinetic energy	SST and Chien k-ϵ turbulence models
`omega`	Specific dissipation rate	SST turbulence model
`epsilon`	Turbulent dissipation	Chien k-ϵ turbulence model

Continued on next page

Table 2: `HISTORY` Keyword — `q` Variables (*Continued*)

Name	Definition	When Used
—*Chemistry equations*		
species	Mass fraction of *species*, where *species* is the species name defined in the chemistry data file (see Section 7.10 and the `CHEMISTRY` keyword block)	Non-reacting chemistry
`rho*`*species*		Finite-rate reacting chemistry
`shf`	Specific heat of formation	Chemistry
—*Navier-Stokes equations in rotating frame*		
`rho*ur`	Momentum in the x direction in a rotating reference frame (`rho*u`, etc., are in the inertial frame)	Rotating reference frame
`rho*vr`	Momentum in the y direction in a rotating reference frame	
`rho*wr`	Momentum in the z direction in a rotating reference frame	
`rho*e0r`	Total energy in a rotating reference frame	
—*MFD equations*		
`Bx, By, Bz`	Magnetic field components in the x, y, and z directions	MFD flows
`Ex, Ey, Ez`	Electric field components in the x, y, and z directions	
`Jx, Jy, Jz`	Current density components in the x, y, and z directions	
`Lx, Ly, Lz`	Lorentz force components in the x, y, and z directions	
`sigma`	Conductivity	
`voltage`	Voltage	

`FREQUENCY` *f*

Specifies the number of solver iterations between output of time history information. The default is to write to the *.cth* file every 10 iterations.

```
REGION nzn ibeg iend jbeg jend kbeg kend
REGION nzn NODE nodenum
REGION nzn CELL cellnum
...
```

Defines the discrete locations where temporal information will be sampled in zone *nzn*. The three formats correspond to a structured zone range, a particular node in an unstructured node-

based scheme, and a particular cell in an unstructured cell-based scheme, respectively.[25]

Except for the special-purpose variables `radius` and `DeltaP`, in a structured grid time history data will be recorded at each point in the domain defined by the indices (*ibeg,jbeg,kbeg*) to (*iend,jend,kend*). The variables `radius` and `DeltaP` are intended for use in tracking the location of a shock wave. In this case, the `REGION` parameters are defined differently. See the section Front Tracking Mode for details.

Multiple `REGION` lines may be used to specify multiple sampling regions, in the same zone, or different zones. Grid sequencing is supported for structured grids, but there is presently no other way (except multiple `REGION` lines) to spatially sub-sample a region.

`ENDHISTORY`

Defines the end of the `HISTORY` block.

Front Tracking Mode

For structured grids, a special mode exists that can be useful for tracking the motion of shock waves. If the region is specified as:

`REGION` *nzn i0 idir j0 jdir k0 kdir*

where one of *idir*, *jdir*, *kdir* is ± 1 and the other two are zero, then front tracking mode is active in zone *nzn*.

Beginning at the point (*i0,j0,k0*), the solution is sampled to the end of the zone in the specified direction, looking for the maximum change in pressure across a cell. This location is then used as the "probe location" to output the specified variables. In this context, the `radius` variable represents the distance of this point from the point (*i0,j0,k0*). Note that this is probably the only time that a time history variable of x, y, or z makes sense, as they can be used to more precisely specify the location of the detected front.

To provide an indication of the evolution of the shock strength, the variable `DeltaP` can be used. When this variable is specified, the search is continued from the detected front location to the end of the zone, in the specified search direction, and the maximum monotone pressure change is recorded.

Example 1

The following use of the `HISTORY` block will create a time history (*.cth*) file containing the static pressure at the points (17,1,1), (17,11,5), and (9,11,7) in zones 1, 2, and 3, respectively. Values will be written to the file every ten iterations.

```
History
   Variable p
   Region 1 17 17  1  1 1 1
   Region 2 17 17 11 11 5 5
   Region 3  9  9 11 11 7 7
Endhistory
```

Example 2

In the following example, the Mach number and pressure will be recorded in zone 5, for the grid points $(i, j, k) = (7\text{–}16,22\text{–}24,19)$. Values will be written to the file every five iterations.

```
History
```

[25] Note, though, that the production version of WIND does not yet support unstructured grids.

```
   Variable M p
   Frequency 5
   Region 5 7 16 22 24 19 19
Endhistory
```

Example 3

In this example, the position and strength of a shock wave will be tracked in the $+i$ direction in zone 4. The point (1,52,20) will be used as the reference point, and values will be recorded every 20 iterations.

```
History
   Variable radius x DeltaP
   Frequency 20
   Region 4 1 1 52 0 20 0
Endhistory
```

HLLE — HLLE scheme anti-diffusion terms

```
HLLE EPSS epss [zone_selector]
```

This keyword allows the user to control the behavior of the anti-diffusion term in the HLLE scheme. The parameter *epss* is used in the computation of a switch used to regulate the anti-diffusion terms. The possible values for *epss* are shown below.

epss	Switch	Result
0	1.0	Anti-diffusion terms are always fully on. This is the default.
> 0	$1.0 - epss\|p_{scal}\|$	Anti-diffusion terms are on, scaled by the absolute value of a pressure gradient scaling parameter p_{scal}, with *epss* as a coefficient. The result is that the anti-diffusion terms are smaller in regions where the pressure gradient is changing rapidly. Larger values of *epss* result in smaller anti-diffusion terms. The switch is limited to a minimum value of 0.0 (fully off).
< 0	$1.0 + epss$	Anti-diffusion terms are on, with the switch hard-wired to the indicated value. Larger negative values of *epss* result in smaller anti-diffusion terms. The switch is limited to a mimimum value of 0.0 (fully off), which corresponds to $epss = -1.0$.

See Also: RHS

HOLD — Hold conditions at freestream inflow boundaries

```
HOLD {TOTALS | CHARACTERISTICS} [zone_selector]
```

Specifying `HOLD TOTALS` indicates that total conditions are to be held constant at freestream boundaries with subsonic inflow, and is only valid for an ideal gas. Specifying `HOLD CHARACTERISTICS` indicates that characteristic values are to be held constant. Note that the total pressure is always held fixed at freestream inflow boundaries, whether `HOLD TOTALS` is specified or not. The option specified will be applied at all the freestream boundaries in the specified zone(s).

In WIND 5.201 and later the default is `HOLD CHARACTERISTICS`. In earlier versions the default is `HOLD TOTALS`.

Note that the `HOLD` keyword only applies to freestream boundaries with subsonic inflow. See the `HOLD_TOTALS` keyword in the `ARBITRARY INFLOW` keyword block for information on holding total conditions at arbitrary inflow boundaries.

Note also that the syntax is slightly different for freestream and arbitrary inflow boundaries. For freestream boundaries, `HOLD TOTALS` and `HOLD CHARACTERISTICS` are used, without an underscore. For arbitrary inflow boundaries, `HOLD_TOTALS` and `HOLD_CHARACTERISTICS` are used in the `ARBITRARY INFLOW` keyword block, with an underscore.

IMPLICIT — Implicit operator control

```
IMPLICIT {word1 [word2 word3 [zone_selector [tre]]] | \
          JACOBI jiter jcnv [RELAX jrel] [zone_selector] | \
          GAUSS giter gcnv [RELAX grel] [zone_selector] | \
          MACCORMACK [MAFK k] [zone_selector] | \
          OVERFLOW [zone_selector]}
```

This keyword allows user control of the implicit operator within each zone. There are five possible modes, defined as follows:

```
IMPLICIT word1 [word2 word3 [zone_selector [tre]]]
```

This form of the the `IMPLICIT` keyword allows the user to turn off the implicit operator completely, resulting in an explicit calculation, or to use a scalar or full block implicit operator. With these options, a different implicit operator may be specified for each computational direction.

The user-specified parameters are defined as follows:

word1 A keyword controlling the ξ-direction implicit operator

word2 A keyword controlling the η-direction implicit operator

word3 A keyword controlling the ζ-direction implicit operator

tre Specifies the trapezoidal time differencing factor. The possible values are:

0	Explicit (but expensive, don't use)
1 or omitted	Implicit
0.5	Trapezoidal time differencing

The keywords that may be used for *word1*, *word2*, and *word3* are defined in the following table.

Keyword	Difference Scheme
`NONE`	Explicit
`SCALAR`	Scalar implicit operator (default for Euler solutions)
`FULL`	Full block implicit operator (default for viscous solutions)

Note that if only one of the above keywords is specified, WIND assumes that the specified operator is to be used in all directions in all zones.

```
IMPLICIT JACOBI jiter jcnv [RELAX jrel] [zone_selector]
```

This form of the `IMPLICIT` keyword turns on the point Jacobi implicit operator in the selected zone. The user-specified parameters are defined as follows:

jiter The number of subiterations allowed each time step. A typical value is 30.

jcnv The level of convergence to assume the subiterations are converged. A typical value is 0.0001.

jrel The relaxation factor. The default value is 1.0 (i.e., no relaxation).

`IMPLICIT GAUSS` *giter gcnv* `[RELAX` *grel*`]` `[`*zone_selector*`]`

This form of the `IMPLICIT` keyword turns on the Gauss-Seidel implicit operator in the selected zone. The user-specified parameters are defined as follows:

giter The number of subiterations allowed each time step. A typical value is 10.

gcnv The level of convergence to assume the subiterations are converged. A typical value is 0.0001.

grel The relaxation factor. The default value is 1.0 (i.e., no relaxation).

`IMPLICIT MACCORMACK [MAFK` *k*`]` `[`*zone_selector*`]`

This form of the `IMPLICIT` keyword turns on MacCormack's first-order modified approximate factorization (MAFk) procedure in the selected zone. The procedure attempts to remove the decomposition error by feeding the error term back into the matrix equations on the right-hand side.

The solution process requires subiterations, with the number of subiterations specified with the `MAFK` option by the parameter *k*. The default for *k* is 2.

For stability, the `IMPLICIT BOUNDARY` keyword should be used to specify that implicit boundary conditions are to be used with the MAFk procedure. In addition, when large CFL numbers are to be used, it is recommended that the CFL number be increased gradually (over 200 iterations or so) to the desired value using the `INCREMENT` parameter in the `CFL#` keyword.

The MAFk procedure has been demonstrated in WIND to be stable in two dimensions with very high CFL numbers (greater than 1000). In three dimensions, however, only limited testing has been done, and its efficiency has not yet been determined.

`IMPLICIT OVERFLOW [`*zone_selector*`]`

This form of the `IMPLICIT` keyword specifies that the "ARC3D 3-factor diagonal scheme" as implemented in OVERFLOW 1.8q is to be used. Tests indicate that this scheme is faster than the other implicit schemes, and gives comparable answers.

This option is currently available only for 3-d perfect gas flows, with explicit boundary conditions. It also uses more memory than the other implicit schemes, due to the interface coding used to implement it in the WIND code.

See Also: `IMPLICIT BOUNDARY`

IMPLICIT BOUNDARY — Implicit boundary conditions

```
IMPLICIT BOUNDARY {ON | OFF} [zone_selector]
```

This keyword controls the use of WIND's numerical boundary conditions within the implicit operator. The default mode is `OFF`, implying explicit boundary conditions. When `ON` is specified, implicit boundary conditions will be used at "wall"-type boundaries.

Implicit boundary conditions cannot currently be used if the `IMPLICIT OVERFLOW` keyword is specified.

See Also: `IMPLICIT`

INCLUDE — Include a file in the standard input

```
INCLUDE filename
```

This keyword allows the contents of an external file to be incorporated into WIND's standard input, just as if they were part of the input data (*.dat*) file. The *filename* must be either the full path name for the file, or a relative path name within the directory in which the WIND code is being run.

"Include" files may be nested.

ITERATIONS — Set number of iterations per cycle

```
{ITERATIONS [PER] [CYCLE] | ITER_CYCLE} n [PRINT FREQUENCY freq] \
   [zone_selector]
```

This keyword allows the user to control the number of iterations performed in each zone. The parameter n specifies the number of iterations per cycle (the default value is 5). In addition, n may have the following special values:

0 The indicated zone(s) will be bypassed, but the zone coupling will still take place. This is not a good idea as the active zone will update inactive ($n = 0$) zones, overwriting the existing data. This inactive zone will then pass the active zone's data back to it.

−1 Perform no iterations in the indicated zone and *do not* update this zone with adjacent zone information (but *do* pass this zone's frozen information to other zones each cycle).

−2 Perform no iterations in the indicated zone and *do not* update this zone with adjacent zone information (and *do not* pass this zone's frozen information to other zones).

Residuals will be written to the *.lis* file every *freq* iterations, but only for cycles consistent with the `PRINT FREQUENCY` specified with the `CYCLES` keyword. The default value for *freq* is 1.

Note that this does not affect integrated properties specified using the `LOADS` keyword block. The print frequency for integrated properties is controlled by the `PRINT` keyword in the `LOADS` keyword block.

The *zone_selector*, if specified, must appear last. The print frequency, however, is global; the value specified on the last `ITERATIONS` entry in the input data (*.dat*) file will be used in all zones.

Example

If the user specifies

```
CYCLES 1000 PRINT FREQUENCY 10
ITERATIONS PER CYCLE 10 PRINT FREQUENCY 5
```

cycle time information will be printed for cycles 1, 11, 21, etc. Residuals will be printed for iterations 10 (in cycle 1), 110 (cycle 11), 210 (cycle 21), etc.

See Also: `CYCLES`, `LOADS`, `NAVIER-STOKES ITERATIONS`, `TURBULENCE`

LOADS — Flowfield integration (block)

```
LOADS
   [PRESSURE [OFFSET] {FREESTREAM | val}]
   PRINT [PLANES] [TOTALS] [ZONES] [LIFT | DRAG] [FREQUENCY freq] [MAKE_FRC file]
   [REFERENCE AREA aref]
   [REFERENCE LENGTH lref]
   [REFERENCE MOMENT CENTER xc yc zc]
   ZONE nzn
   SUBSET I range J range K range word1 [word2 [word3...]]
   SURFACE {I val | J val | K val} word1 [word2 [word3...]]
ENDLOADS
```

Flowfield properties may be integrated during the course of a WIND run to check convergence and solution quality. The values printed to the list file represent the force coefficients $F/A_r q_r$ and moment coefficients $M/L_r A_r q_r$ in the x, y, and z coordinate directions, where F and M are the force and moment, and L_r, A_r, and q_r are the reference values of length, area, and dynamic pressure. Integrated mass and momentum fluxes may also be computed. The values are written into the list output (*.lis*) file.

The various elements of the `LOADS` input block are defined as follows:

```
PRESSURE [OFFSET] {FREESTREAM | val}
```

This keyword controls the pressure integration. By default, the code uses $P - P_\infty$ in all pressure integrations, where P_∞ is the static pressure at freestream conditions. This keyword permits the user to specify the pressure offset value, such that:

`FREESTREAM`	Use $P - P_\infty$
val	Use $P - val$

```
PRINT [PLANES] [TOTALS] [ZONES] [LIFT | DRAG] [FREQUENCY freq] [MAKE_FRC file]
```

This keyword controls the output data from the integration. Options are turned 'on' by including the appropriate keyword.

`PLANES`	Output the result of the integration for each subset specified.
`TOTALS`	Output integration grand totals over all the zones at the end of each cycle.
`ZONES`	Output integration totals for each zone.
`LIFT | DRAG`	Output lift, drag, and side forces instead of (x, y, z) force components. Directions for the lift, drag, and side force components are computed from the angles of attack and yaw specified using the `FREESTREAM` keyword.
`FREQUENCY`	Output integration results (except for grand totals) every *freq* iterations.
`MAKE_FRC`	Write grand totals to the summary file *file* at the end of each cycle, in addition to the *.lis* file. This requires that `PRINT TOTALS` also be specified.

```
REFERENCE AREA aref
REFERENCE LENGTH lref
REFERENCE MOMENT CENTER xc yc zc
```

These keywords specify the reference area *aref* in square inches; the reference length *lref* in inches; and the coordinates (*xc*,*yc*,*zc*) of the reference moment center in inches. The default values are 1.0 for *aref*, 1.0 for *lref*, and (0.0, 0.0, 0.0) for (*xc*, *yc*, *zc*).

```
ZONE nzn
```

This keyword specifies the zone number *nzn* for the subsets which follow.

```
SUBSET I range J range K range word1 [word2 [word3...]]
```

The *range* parameters define a subset of a computational surface in zone *nzn* over which the integration is to be performed, and take one of the following forms:

index1 index2	Starting and ending indices in the specified direction. `LAST` may be used for the last index.
`ALL`	Equivalent to `1 LAST`.

For three-dimensional cases, the starting and ending indices for one (and only one) of the `I`, `J`, or `K` parameters must be the same. For two-dimensional cases, the `K` parameter must be specified as either `K 1 1` or `K ALL`; one or both of the `I` and `J` parameters may have different starting and ending indices.

The *word* parameters are keywords specifying what flowfield properties are to be computed, plus a couple of additional control values. The following keywords are currently available:

`FORCE`	Compute pressure forces.
`MASS`	Compute mass flow.
`MOMENT`	Compute moments.
`MOMENTUM`	Compute momentum.
`VISCOUS`	Compute viscous force and moments.
`NORMAL` *inorm*	*inorm* $= \pm 1$, specifying the desired normal vector direction; $+1$ for the increasing index direction and -1 for the decreasing index direction. *inorm* defaults toward the interior of the grid if the integration surface is on a boundary of the zone.
`NOSLIP`	Integrate only over points where the total velocity is 0.

```
SURFACE {I val | J val | K val} word1 [word2 [word3...]]
```

This keyword is similar to the `SUBSET` keyword, except that it provides a slightly simpler syntax for specifying a complete coordinate surface in zone *nzn* over which the integration is to be performed. The parameter *val* is a coordinate index specifying the surface.

The word parameters are the same as those described above for the `SUBSET` keyword.

Example

```
LOADS
   PRESSURE OFFSET 0.0
   PRINT PLANES ZONES TOTALS LIFT FREQUENCY 2
   REFERENCE AREA 100.
   REFERENCE LENGTH 12.
   REFERENCE MOMENT CENTER 35. 36. 78.
   ZONE 4
      SUBSET I ALL J 1 1 K 15 LAST FORCE MOMENT VISCOUS NOSLIP
   ZONE 10
      SURFACE I 1 MOMENTUM MASS
ENDLOADS
```

MARCHING — Parabolized Navier-Stokes algorithm

```
MARCHING [LIMITER value] [CHECKPOINT interval] [COPY]
```

This keyword enables WIND's spatial marching, or parabolized Navier-Stokes (PNS), algorithm for flowfields which are supersonic in the computational i-direction. In this mode, WIND marches from the $i = 1$ to the $i = i_{max}$ computational surface, attempting to compute a steady-state solution at each plane before moving on to the next one. Using the PNS algorithm significantly reduces the computing time required for supersonic solutions.

`LIMITER` *value* — This keyword enables the marching limiter, which limits the change in the solution vector $\mathbf{Q}$ to $(value)\mathbf{Q}$. I.e.,

$$\Delta\mathbf{Q} \leq (value)\mathbf{Q}$$

`CHECKPOINT` *interval* — The parameter *interval* specifies the number of i-planes to be computed before writing the current flowfield to the solution file. The default is 10.

`COPY` — This keyword requests that WIND copy the solution from the most recently computed i-plane to the upcoming i-plane, giving a (hopefully) better initialization to the new i-plane than simply starting from freestream flow.

Notes

- Marching is only available when WIND is run in serial mode.
- Marching is not available with the one-equation turbulence models (`BARTH`, `SPALART`).
- Marching requires that one of the following explicit operators be used. (See the `RHS` keyword.)
  - Coakley (any order)
  - Roe (first-order upwind, second-order upwind, third-order upwind-biased, first-order upwind modified for stretched grids, or second-order upwind-biased modified for stretched grids)
  - Van Leer (first-order upwind, second-order upwind, or third-order upwind-biased)
- The Roe, Van Leer, and HLLE second-order upwind-biased explicit operators modified for stretched grids use the $i + 1$ grid point, which is invalid in a PNS solution. When these operators are used, WIND automatically changes to first order in the i-direction.

See Also: `RHS`, `TURBULENCE`

MASS FLOW — Outflow boundaries, mass flow

```
MASS [FLOW] {RATE [ACTUAL | CORRECTED] | RATIO} value \
   [PRESSURE | DIRECT [RELAXER rlxr]] [ORDER {ZERO|0|ONE|1}] \
   [ZONE] range1[,range2[, ...  rangen]]
```

This keyword allows the user to specify mass flow at outflow boundaries in the flowfield. The specified *value* must be positive.

`RATE` *value* represents the mass flow rate in $\mathrm{lb_m}$/sec, and may be actual (the default) or corrected, as specified by the `ACTUAL` or `CORRECTED` keyword. The corrected air flow is defined as

$$W_c = W_{actual}\frac{\theta_x^{0.5}}{\delta_x}$$

where

$$\delta_x = P_x/P_0 \qquad \theta_x = T_x/T_0$$

and P_x and T_x are the total pressure and temperature at the duct exit, and P_0 and T_0 are equal to 14.7 psi and 520 °R, respectively.

`RATIO` *value* represents the mass flow ratio. The actual mass flow is computed as

$$\dot{m} = (value)\rho_\infty U_\infty A_{cap}$$

where A_{cap} is the capture area found in the *.cgd* file zonal parameters.

`PRESSURE` A spatially-constant pressure is set at the boundary, and modified as the solution proceeds until the desired mass flow is achieved. This is the default.

`DIRECT` The momentum, and thus the mass flow, is modified directly, and the pressure adjusts as the solution proceeds.

`RELAXER` The specified mass-flow rate will be relaxed using the relaxtion factor *rlxr*. This option only applies when `DIRECT` is specified. The default value for *rlxr* is 1.0 (i.e., no relaxation).

`ORDER` Either zeroth- or first-order extrapolation will be used, as specified. The default is zeroth-order.

In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone`:`*endzone*	Selects all zones from *begzone* to *endzone*
`ALL`	Selects all zones

Note that if multiple zones are listed, the specified mass flow will be applied in *each* zone.

With the `PRESSURE` option, the pressure will be constant over the entire outflow boundary, resulting in poor solutions for flows that should have cross-flow pressure gradients in that region. With the `DIRECT` option, cross-flow pressure gradients may be present at the outflow boundary, and the mass flow will be equal to the user-specified value (for $rlxr = 1$) for all iterations.

For flows with negligible cross-flow pressure gradients, the results and convergence rates using the `PRESSURE` and `DIRECT` options are nearly the same. For a test case with a significant cross-flow pressure gradient near the outflow boundary, the computed pressures using the two options differed by as much as 10%. The `PRESSURE` option, although non-physical for this case, had a slightly better convergence rate.

Internally, to apply this boundary condition WIND does the following:

1. At all the boundary points, the density, momentum, and energy are extrapolated from the interior to the boundary using either zeroth- or first-order extrapolation, as specified.
2. The mass flow at the boundary is computed by numerical integration.
3. If `PRESSURE` was specified, a new downstream pressure is computed based on the difference between the computed and specified mass flow rates.
4. If `DIRECT` was specified, a mass flow correction factor is computed as

$$f_{corr} = 1 + r\left(\frac{\dot{m}_{spec}}{\dot{m}_{int}} - 1\right)$$

   where r is the input relaxation factor *rlxr*, and $\dot{m}_{spec}$ and $\dot{m}_{int}$ are the user-specified and integrated mass flow rates, respectively.
5. For each boundary point, the density and momentum, plus the pressure, effective gamma, compressibility factor, and speed of sound, are extrapolated from the interior to the boundary using either zeroth- or first-order extrapolation, as specified.
6. If `DIRECT` was specified, the extrapolated momentum values at the outflow boundary are modified using

$$\begin{aligned}(\rho u) &= f_{corr}(\rho u)_{ext}\\ (\rho v) &= f_{corr}(\rho v)_{ext}\\ (\rho w) &= f_{corr}(\rho w)_{ext}\end{aligned}$$

7. If `PRESSURE` was specified, the energy at each boundary point is computed, consistent with the downstream pressure value from step 3 and the extrapolated values of density, etc., from step 5.
8. If `DIRECT` was specified, the energy at each boundary point is computed, consistent with the momentum values from step 6 and the extrapolated values of density, pressure, etc., from step 5.

Extrapolation Notes

The default for all extrapolation is zeroth-order (i.e., conditions at the boundary are set to the values at the computational plane adjacent to the boundary). This results in a discontinuous slope in flow values near the outflow boundary, which may be important for flows with significant streamwise pressure gradients. First-order extrapolation yields smoother results.

For flows with little or no streamwise pressure gradient near the outflow boundary, the results using zeroth- and first-order extrapolation are essentially identical. Convergence rates and the final residual values are generally better with zeroth-order extrapolation, however, so the default zeroth-order extrapolation is recommended.

For flows with significant streamwise pressure gradients near outflow boundaries, zeroth-order extrapolation can give poor results at the outflow boundary, and in some cases these can affect values at the inflow boundary. First-order extrapolation is thus recommended for these flows.

Examples

```
MASS FLOW RATIO           0.95 ZONE 2
MASS FLOW RATE  ACTUAL    180. ZONE 3
MASS FLOW RATE  CORRECTED 220. ZONE 4
```

See Also: COMPRESSOR FACE, DOWNSTREAM PRESSURE, DOWNSTREAM MACH

MFD — Magneto-Fluid Dynamics Model (block)

```
MFD
   [OUTPUT {BFIELD | CONDUCTIVITY | CURRENT | EFIELD | VOLTAGE | LORENTZ}]
   [RELAX_MFD nriter]
   [UPDATE nuiter]
   [RADIATION emiss lref [tback]]
   {LORENTZ FILE fn FREQUENCY f [DUTY du] [SCALE sc] \
            PHASES n PATTERNS p1 p2 p3 ... pn
    |
    {CONDUCTIVITY | SIGMA} {CONSTANT sigma | \
                           CFL | \
                           EQUILIBRIUM {ARGON | AIR | GAS} [POTASSIUM mk] | \
                           LINEAR t1 sig1 t2 sig2 | \
                           PREDICTED [USING] [LIN-RESSLER | BOEING]}
    BFIELD {CONSTANT bz | BLOCKS nbblocks | CFL}
    {EFIELD {CONSTANT ey | BLOCKS neblocks | CFL} | \
     VOLTAGE {BOUNDARIES nvbnd | CFL | PARAMETERS mitvlt vlttol vltrx vltry vltrz vltfac}
    }
   }
ENDMFD
```

The MFD keyword block allows the user to specify the magneto-fluid dynamics mode and input data for the desired forms of the magnetic and electric fields, the electrical conductivity, output variables, and approximate radiation energy losses.

Control Functions

`MFD`

Defines the beginning of the MFD block.

`ENDMFD`

Defines the end of the MFD block.

`OUTPUT {BFIELD | CONDUCTIVITY | CURRENT | EFIELD | VOLTAGE | LORENTZ}`

The specified data will be written into into the flow (*.cfl*) file. Multiple `OUTPUT` keywords may be specified, to write multiple types of data into the *.cfl* file.

`RELAX_MFD` *nriter*

The MFD source terms will be relaxed over *nriter* iterations. The default for *nriter* is 1.

`UPDATE` *nuiter*

The MFD source terms will be updated every *nuiter* Navier-Stokes iterations. The default for *nuiter* is 1.

```
RADIATION emiss lref [tback]
```

Estimate the energy loss due to thermal radiation of the fluid with emissivity *emiss*, optical depth *lref*, and background temperature *tback* (°R). The default for *tback* is the freestream static temperature.

Body Force Determination

The body force resulting from the MFD terms can be added in one of two ways: (1) by directly specifying the Lorentz force; or (2) by specifying the conductivity, the magnetic field, and either the electric field or voltage.

The following keyword is used to directly specify the Lorentz force. If this method is used, the CONDUCTIVITY, BFIELD, EFIELD, and VOLTAGE keywords are not allowed.

```
LORENTZ FILE fn FREQUENCY f [DUTY du] [SCALE sc] \
        PHASES n PATTERNS p1 p2 p3 ... pn
```

nl Lorentz force distributions are stored in a *.cem* file. This distribution is spread over space depending upon the phasing patterns. The pattern changes *n* times during a cycle.

FILE *fn*	Specifies the name of the *.cem* file containing the Lorentz force patterns
FREQUENCY *f*	Number of cycles/second
DUTY *du*	Force is on the first *du* portion of a phase. The default value is 1.0.
SCALE *sc*	Multiply the Lorentz force by *sc* before adding it to the equations. The default value is 1.0.
PHASES *n*	Number of pattern phases in a cycle
PATTERNS *p1 p2 p3 ... pn*	The Lorentz force distribution to use in each phase. Must correspond to the *nl* distributions stored in the *.cem* file.

If the Lorentz force is not directly specifed using the LORENTZ keyword, you must use the CONDUCTIVITY and BFIELD keywords to specify the conductivity and the magnetic field, and either the EFIELD or VOLTAGE keywords to specify the electric field or voltage.

```
{CONDUCTIVITY | SIGMA} {CONSTANT sigma | \
                       CFL | \
                       EQUILIBRIUM {ARGON | AIR | GAS} [POTASSIUM mk] | \
                       LINEAR t1 sig1 t2 sig2 | \
                       PREDICTED [USING] [LIN-RESSLER | BOEING]}
```

This keyword specifies the electrical conductivity in mhos/meter.

CONSTANT *sigma*	Hold the conductivity constant at the value *sigma*
CFL	Read the conductivity from the flow (*.cfl*) file

`EQUILIBRIUM {ARGON | AIR | GAS} [POTASSIUM` *mk*`]`
: Estimate the electron density as a function of temperature for the indicated gas as input to the Lin & Ressler conductivity model. The default gas is air. If `POTASSIUM` *mk* is specified, the effect of the mass fraction *mk* of potassium will be included.

`LINEAR` *t1 sig1 t2 sig2*
: Set the conductivity to *sig1* at and below the temperature *t1*; to *sig2* at and above the temperature *t2*; and use a linear distribution for temperatures between *t1* and *t2*. The temperatures are in °R.

`PREDICTED [USING] [LIN-RESSLER | BOEING]`
: Use real-gas predicted electron densities for input to the indicated conductivity model. The default is the Lin & Ressler model.

`BFIELD {CONSTANT` *bz* `| BLOCKS` *nbblocks* `| CFL}`

This keyword specifies the magnetic field in tesla.

`CONSTANT` *bz*
: Hold the magnetic field constant at the value *bz*, in the *z* coordinate direction

`BLOCKS` *nbblocks*
: Specify the magnetic field by reading in *nbblocks* blocks of data containing the magnetic field vector at selected coordinate points. The data immediately follows the `BFIELD BLOCKS` keyword. See the **Field Block Description** for details and an example.

`CFL`
: Read the magnetic field from the flow (*.cfl*) file

`EFIELD {CONSTANT` *ey* `| BLOCKS` *neblocks* `| CFL}`

This keyword specifies the electric field in Volts/meter.

`CONSTANT` *ey*
: Hold the electric field constant at the value *ey*, in the *y* coordinate direction

`BLOCKS` *neblocks*
: Specify the electric field by reading in *neblocks* blocks of data containing the electric field vector at selected coordinate points. The data immediately follows the `EFIELD BLOCKS` keyword. See the **Field Block Description** for details and an example.

`CFL`
: Read the electric field from the flow (*.cfl*) file

`VOLTAGE {BOUNDARIES` *nvbnd* `| CFL | PARAMETERS` *mitvlt vlttol vltrx vltry vltrz vltfac*`}`

With this keyword the electric field is determined by specifying the electric potential.

`BOUNDARIES` *nvbnd*
: Specify the electric potential at *nvbnd* zonal boundary regions. The data immediately follows the `VOLTAGE BOUNDARIES` keyword. See the **Voltage Boundary Description** for details and an example.

`CFL`
: Read the electric potential field from the flow (*.cfl*) file

`PARAMETERS` *mitvlt vlttol vltrx vltry vltrz vltfac*

Iterate a maximum of *mitvlt* iterations (the default is 10,000) to a tolerance of *vlttol* (a positive value means to a level of *vlttol*, a negative value means $|vlttol|$ orders of magnitude; the default is a level of 10^{-12}) with implicit factors *vltrx*, *vltry*, and *vltrz* (currently these must all be set to 0.0) and with an over-relaxation factor of *vltfac* (a typical value is 0.4).

Field Block Description

By using the `BFIELD BLOCKS` and/or `EFIELD BLOCKS` keywords, the magnetic and/or electric field may be determined from blocks containing the field vector at selected coordinate points. Each block consists of eight points in space, with the corresponding field vector at each of those points. The region contained within the blocks is filled using tri-linear interpolation between the specified points. Up to eight blocks may be specified for each field type. The spatial locations of the blocks may overlap, with the later-specified blocks overwriting the earlier ones.

There are eight lines of input per block, one for each of the eight coordinate points. Each line contains six values—the x, y, and z coordinates, and the field vector components in the x, y, and z directions.

The following example specifies the magnetic field using two blocks. (The comments in a slanted font are not part of the input.)

```
BFIELD BLOCKS 2
    60.0    0.0   0.0    0.0   0.0   0.0     Upstream plane of block 1
    60.0   20.0   0.0    0.0   0.0   0.0
    60.0    0.0   1.0    0.0   0.0   0.0
    60.0   20.0   1.0    0.0   0.0   0.0
   120.0    0.0   0.0    0.0   0.0  10.0     Downstream plane of block 1
   120.0   20.0   0.0    0.0   0.0   5.0
   120.0    0.0   1.0    0.0   0.0  10.0
   120.0   20.0   1.0    0.0   0.0   5.0
   120.0    0.0   0.0    0.0   0.0  10.0     Upstream plane of block 2
   120.0   20.0   0.0    0.0   0.0   5.0       (same as downstream of 1)
   120.0    0.0   1.0    0.0   0.0  10.0
   120.0   20.0   1.0    0.0   0.0   5.0
   180.0    0.0   0.0    0.0   0.0   0.0     Downstream plane of block 2
   180.0   20.0   0.0    0.0   0.0   0.0
   180.0    0.0   1.0    0.0   0.0   0.0
   180.0   20.0   1.0    0.0   0.0   0.0

     x      y     z      Bx    By    Bz
```

Voltage Boundary Description

Voltage boundaries are used to specify the boundary conditions at zone boundaries for the electric potential solver. The electric potential solver will take into account gradients and discontinuities in conductivity, as well as the electromotive force (EMF) induced by the fluid motion through the magnetic field. Currently, accurate voltage predictions require a grid with low skewness.

Up to 64 boundaries may be specified. The conditions at each boundary are specified on a single line, with eight values—*zone*, *face*, *type*, *L1*, *L2*, *M1*, *M2*, *value*—defined as follows:

zone Zone containing the boundary

face The boundary face, specified as a number from 1 to 6 corresponding to the i_1, i_{max}, j_1, j_{max}, k_1, and k_{max} face, respectively

type The boundary type, specified as 0 for a reflection boundary, and 1 for specified voltage

L1, L2, M1, M2 The indices on the face over which the boundary condition applies, as follows

Face	Indices
i_1 or i_{max}	j_{low}, j_{high}, k_{low}, k_{high}
j_1 or j_{max}	k_{low}, k_{high}, i_{low}, i_{high}
k_1 or k_{max}	i_{low}, i_{high}, j_{low}, j_{high}

value The voltage for specified-voltage boundaries; 0.0 for reflection boundaries

The following example specifies eight voltage boundaries. (The comments in a slanted font are not part of the input.)

```
VOLTAGE BOUNDARIES 8
   1     1     0     1    30    1     1       0.0
   1     1     1    31    61    1     1       0.0
   1     2     1     1    31    1     1   10000.0
   1     2     0    32    61    1     1       0.0
   1     3     0     1     1    1    30       0.0
   1     3     1     1     1   31    61   10000.0
   1     4     1     1     1    1    31       0.0
   1     4     0     1     1   32    61       0.0

 Zone  Face  Type   L1    L2   M1    M2     Value
```

MOVING WALL — Specify moving wall boundaries

```
MOVING [WALL] iblreg {[TRANSLATE {I | X} speed] | [SPIN X rps yc zc]}
```

This keyword allows no-slip walls to have a tangential velocity.

iblreg Moving walls are activated in regions specified as bleed in the *.cgd* file (set with GMAN). *iblreg* is the bleed region number.

`TRANSLATE I` Move wall in the i-direction at the velocity *speed*, in feet per second.

`TRANSLATE X` Move wall in the x-direction at the velocity *speed*, in feet per second.

`SPIN X` Move wall to model rotation about the x-axis (wall should be spinning and tangent to this motion).

- *rps* Rotation speed in radians per second
- *yc,zc* Coordinates of the center of rotation in the y-z plane

NAVIER-STOKES ITERATIONS — Navier-Stokes sub-iterations

`NAVIER-STOKES ITERATIONS` *iter* [*zone_selector*]

The WIND code organizes the equations to be solved into logical "groups" that are solved together. It also allows multiple iterations of a specific group (i.e., sub-iterations) for each "iteration per cycle". (The number of iterations per cycle is set using the `ITERATIONS` keyword.)

The `NAVIER-STOKES ITERATIONS` keyword allows the user to specify the number of sub-iterations for the Navier-Stokes equation group (which includes any chemical species equations) performed in each zone for each "iteration per cycle". The default value is one, indicating that each "iteration per cycle" corresponds to one iteration of the Navier-Stokes equations. Note that if *iter* is set to zero, the Navier-Stokes equations will not be solved at all.

See Also: `ITERATIONS`, `TURBULENCE`

NEWTON — Use Global Newton time stepping

```
NEWTON [TIME LEVELS newstp] [CONVERGE {LEVEL | ORDER} newcvg]
```

This keyword specifies that the Global Newton time stepping algorithm is to be used. The parameter *newstp* specifies the number of Global Newton time levels to advance. The default value is 30.

There are two possible procedures for determining the overall global convergence.

`CONVERGE LEVEL` Convergence is assumed when

$$|\mathbf{Q}^n - \mathbf{Q}^{n-1}| < newcvg$$

where $\mathbf{Q}$ represents the vector of dependent variables, and n is the Newton time level.

`CONVERGE ORDER` Convergence is assumed when

$$\frac{|\mathbf{Q}^n - \mathbf{Q}^{n-1}|}{\max(|\mathbf{Q}^n - \mathbf{Q}^{n-1}|)} < 10^{-newcvg}$$

The default is `CONVERGE ORDER 3`.

Within a Newton time level, the number of cycles and iterations to be run is specified using the `CYCLES` and `ITERATIONS` keywords.

The convergence criteria within a Newton time level may be specified using the `CONVERGE` keyword. Note that the default for the `CONVERGE` keyword is a four-order of magnitude decrease in the maximum residual. If Newton iteration is being used for a steady flow problem, with the default of three orders of magnitude for the global convergence criteria, it would make sense to also use three orders of magnitude for the convergence criteria within a Newton time step.

See Also: `CYCLES`, `ITERATIONS`, `CONVERGE`

OUTFLOW NON-REFLECTING — Outflow boundaries, non-reflecting

```
OUTFLOW NON-REFLECTING [ZONE] range1[,range2[, ...  rangen]]
```

This keyword imposes a non-reflecting, subsonic outflow boundary condition at outflow boundaries. Acoustic disturbances reaching the boundary are essentially eliminated. It is actually implemented using the Paynter compressor face model (Slater and Paynter, 2000), but with the response coefficients α and β set to zero.

In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
ALL	Selects all zones

This boundary condition is intended to be used for time-accurate flows with a CFL number less than one; however, it may be applicable in certain cases for steady-state simulations.

See Also: COMPRESSOR FACE PAYNTER

PERIODIC — Periodic boundaries

Periodic boundaries are treated as normal coupled boundaries in WIND, with the connection data stored in the grid (*.cgd*) file. No keyword is needed in the input data (*.dat*) file.

GMAN's `CONNECT MODE` option is used to specify how to move one boundary surface to overlay it on the corresponding periodic boundary surface. There are two connection modes — translation and rotation. For translation, the user defines the Δx, Δy, and Δz movement for the surface. For rotation the user sets the center of rotation and the angles. The periodic boundary surfaces do not have to be in the same zone, nor do they have to be point matched. They only have to line up physically once the movement has been performed.

Details about how to set a periodic boundary condition in GMAN are presented in Section 5.6.1 starting on page 50.

REINITIALIZE — Reinitialize selected flowfield zones on restart

```
REINITIALIZE {MODE {AUTO | NOAUTO} | \
              [ZONE] range1[,range2[, ...  rangen]] [IJK_RANGE]}
```

This keyword may be used to reinitialize the flow conditions in specified zones after a restart. The reinitialization will be done as if WIND were being run from scratch, i.e., to freestream values, to values specified using the ARBITRARY INFLOW keyword, and/or to values specified using the BL_INIT keyword, as described in Section 3.6 starting on p. 36.

In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
ALL	Selects all zones

REINITIALIZE may be used in conjunction with the ARBITRARY INFLOW keyword to reinitialize only selected portions of the flow. If the IJK_RANGE parameter is specified with the REINITIALIZE keyword, conditions will be reinitialized only at those points specified via the IJK_RANGE parameter on the ARBITRARY INFLOW keyword. If the IJK_RANGE parameter is not specified with the REINITIALIZE keyword, the conditions at all the grid points in the specified zones will be reinitialized.

Multiple REINITIALIZE keywords are permitted, each on a separate line in the input data file.

If the MODE NOAUTO option is used (the default), and if the grid sizes in the grid (*.cgd*) and flow (*.cfl*) files are different, WIND will stop unless: (1) ITERATIONS PER CYCLE was set to −2 for the zone in question, indicating that the zone plays no part in the calculation; or (2) the user specified that the zone be reinitialized, using REINITIALIZE ZONE ....

If the MODE AUTO option is used, and if the grid sizes in the *.cgd* and *.cfl* files are different, WIND will automatically reset the zone size in the *.cfl* file to match the size in the *.cgd* file, and reinitialize the zone.

See Also: ARBITRARY INFLOW, BL_INIT, FREESTREAM

RELAX COUPLING — Set zone coupling relaxation factor

```
RELAX {COUPLE | COUPLING} factor \
   [ZONE range1[,range2[, ...  rangen]] \
    [[BOUNDARY] {ALL | I1 | IMAX | J1 | JMAX | K1 | KMAX}]]
```

This keyword allows the user to specify the relaxation factor when using characteristic zone coupling. *factor* may range from 0.0 for no coupling, to 1.0 for full zone coupling. In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
begzone:	Selects all zones from *begzone* to MAXZONE, the total number of zones in the grid file
:*endzone*	Selects all zones from 1 to *endzone*
ALL	Selects all zones

If the zone specification is omitted, the specified *factor* will be applied at all boundaries in all zones. Note that a boundary cannot be specified without the zone specification.

If a zone (or zones) is specified, but the boundary is not (or the boundary is specified as BOUNDARY ALL), the specified *factor* will be applied at all boundaries in the specified zone(s).

The default relaxation factor is 0.7 for steady-state calculations, and 1.0 for for space-marching and time-accurate calculations. If the RELAX COUPLING keyword is used, and all zones are selected (either by omitting the zone specification, or by specifying ZONE ALL), the relaxation factor is automatically set to 1.0 for space-marching and time-accurate calculations. However, if specific zones were selected, the specified *factor* will be used at the specified boundary in those zones, even for space-marching and time-accurate calculations.

REL-ROT-ZONE — Relative rotating zones (block)

```
REL-ROT-ZONE
   ZONE iz1 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
        [SUBSET I range J range K range]
   ZONE iz2 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
        [SUBSET I range J range K range]
   ROTATING-ZONE flag
ENDRRZ
```

The REL-ROT-ZONE keyword block, along with the ROTATE keyword, may be used to specify that one zone is rotating relative to another zone. This "relative rotating zone" capability is intended to simulate rotating devices such as compressor fans. The ROTATE keyword is used to specify which zone(s) are rotating, plus the rotation axis and rate. The REL-ROT-ZONE keyword block specifies the location of the interface between the two zones, and how flow conditions are to be transferred between zones.

Zones sharing an interface may have different circumferential extents. Thus, when modeling a turbomachinery component like a compressor, only one blade per stage is required. Multiple zones and zonal interfaces may be used to cover the radial extent of the stage-to-stage interface, but a single zone must be used in the circumferential direction. The interface surface must correspond to a surface of revolution and grid lines in the circumferential direction must be at a constant radius relative to the rotation axis. The rotation axis must correspond to a coordinate axis.

As noted above, this capability is intended to simulate rotating devices such as compressor fans. A typical configuration would be an upstream non-rotating zone covering the full 360° cross-section, and a rotating downstream zone (or zones, if multiple zones are used in the radial direction) with a circumferential extent of $360°/N$ corresponding to a single blade. Periodic boundary conditions would be set at the circumferential boundaries in the rotating zone(s), using GMAN's rotational coupling mode, as described in Section 5.6.1.

The coupling of the downstream face of the non-rotating zone to the upstream face of the rotating zone would also be done using GMAN's rotational coupling mode, repeated $N - 1$ times. This will couple all the points except for that portion of the face that corresponds to the downstream zone in its non-rotated position. These remaining points are then coupled using ordinary (i.e., non-rotated) coupling mode.[26]

The elements of the REL-ROT-ZONE keyword block are defined as follows:

REL-ROT-ZONE

Defines the beginning of the relative rotating zone block.

[26] In WIND, non-zero rotation angles trigger the use of a rotationally periodic boundary condition. Since this is *not* what we want between two relative rotating zones, this "non-rotated" coupling should be done *last*, so that zero rotation angles are written into the common grid (*.cgd*) file.

```
ZONE iz1 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
     [SUBSET I range J range K range]
ZONE iz2 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
     [SUBSET I range J range K range]
```

These two lines define the interface between the two zones. The relevant zones are given by the values of *iz1* and *iz2*, and the relevant boundaries within zones *iz1* and *iz2* are specified via the `BOUNDARY` keyword parameter.

iz1 Zone to which increments will be added when passing information to *iz2*

iz2 Zone receiving positive increments, increments will be subtracted when passing information back to zone *iz1*

The `SUBSET` parameter may be used to specify that the change in properties occurs only over a part of the zone boundary. Otherwise, it is assumed that the change occurs over the entire boundary. The *range* parameters define the part of the zone boundary over which the change occurs, and take one of the following forms:

index1 index2 Starting and ending indices in the specified direction. `LAST` may be used for the last index.

`ALL` Equivalent to `1 LAST`.

The starting and ending indices for the appropriate `I`, `J`, or `K` parameter (depending on the boundary specified) must be the same, and correspond to that boundary.

```
ROTATING-ZONE flag
```

flag defines how data is to be communicated between the two zones. Currently the only valid values are:

0 if the zones are not rotating relative to each other. This is the default.

2 for circumferential averaging. This permits the communication of the bulk fluid properties between zones, while maintaining radial distributions and the efficiency of local time stepping.

```
ENDRRZ
```

Defines the end of the relative rotating zone block.

Example

Figure 8 shows a four-zone configuration. Zone 1 is non-rotating, and zones 2, 3, and 4 are rotating about the x axis at 1680 radians/sec in the counter-clockwise direction. The figure shows the grid from the side in a constant-θ plane, and at the interface plane between the non-rotating and rotating zones, looking downstream.

The non-rotating zone 1 covers the full 360° cross section, but the rotating zones 2–4 cover just 36°. As noted previously, zones sharing an interface may have different circumferential extents.

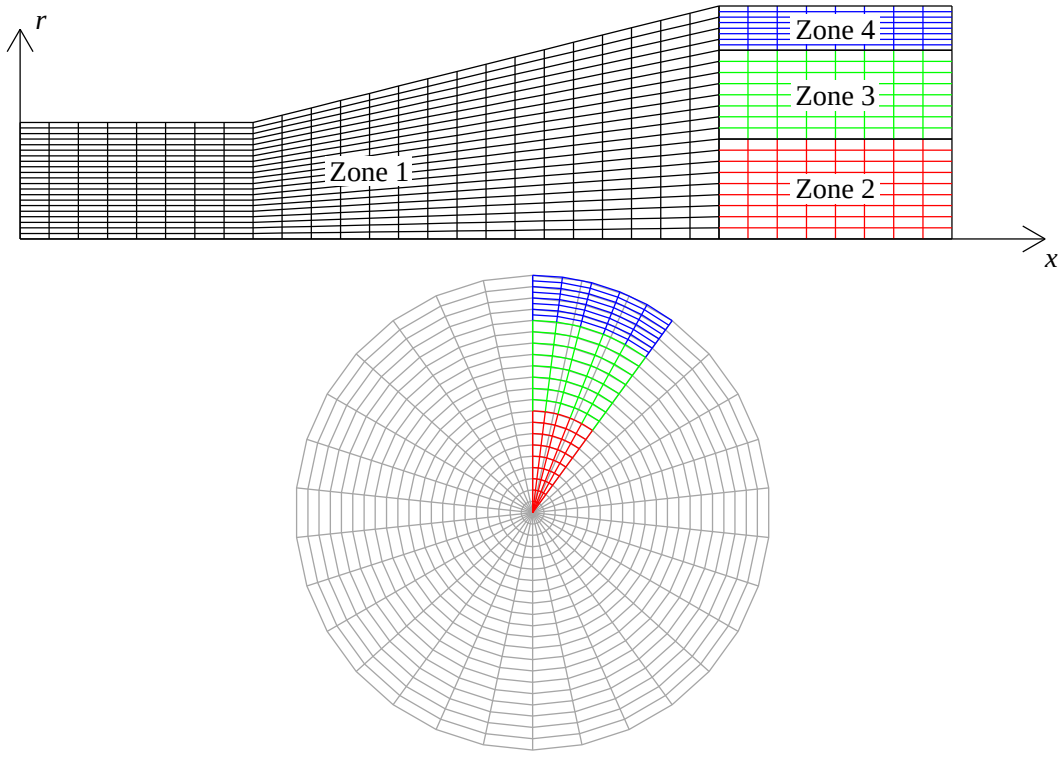


Figure 8: Example grid with relative rotating zones

This particular configuration is similar to one that might be used to model a single blade from a compressor blade row with 10 blades.

The indices i, j, and k are in the axial, radial, and circumferential directions, respectively.

For this configuration, the zone coupling was done in GMAN as described below. In this discussion, the "boundary zone" is the one containing grid points for which a boundary condition is being set. GMAN is used to set the connectivity between points in the boundary zone to points in the "source zone". The source zone is specified using GMAN's "`SEL OTHER BND`" menu choice.

1. Run "auto-couple", which results in the following coupling:

Zone	Boundary		Zone	Boundary
1	`IMAX`	$\Longleftarrow$	2	`I1`
1	`IMAX`	$\Longleftarrow$	3	`I1`
1	`IMAX`	$\Longleftarrow$	4	`I1`
2	`JMAX`	$\Longleftrightarrow$	3	`J1`
3	`JMAX`	$\Longleftrightarrow$	4	`J1`

   Note that "auto-couple" couples the upstream faces of zones 2–4 to the downstream face of zone 1, but not vice-versa.

2. Manually couple the zone 1 `K1` surface ($\theta = 0°$) to the zone 1 `KMAX` surface ($\theta = 360°$), and vice versa, using ordinary coupling.

3. Couple the `K1` and `KMAX` surfaces in zone 2 as periodic boundaries, as described in Section 5.6.1.

(a) Use zone 2, boundary `K1`, as the "boundary zone", and zone 2, boundary `KMAX`, as the "source zone".

(b) Use `1` (rotation) as the connection mode, a point on the x axis as the center of rotation, and `36.0 0.0 0.0` for the rotation angles (in degrees, without commas, and positive for the `+K` direction).

(c) Repeat the above two steps, but using zone 2, boundary `K1`, as the "boundary zone", zone 2, boundary `KMAX`, as the "source zone", and `-36.0 0.0 0.0` for the rotation angles.

(d) Repeat the above three steps for zones 3 and 4.

4. Couple the zone 1 `IMAX` surface to the zone 2 `I1` surface, using rotational coupling mode repeatedly. The procedure is very similar to specifying periodic coupling.

   (a) For the first rotation:

   - In graphics mode, from GMAN's main menu select `BOUNDARY COND`.
   - Pick zone 1, boundary `IMAX` (the "boundary zone").
   - Select `MODIFY BNDY`.
   - Select `COUPLE`.
   - Select `SEL OTHER BND`.
   - Pick zone 2, boundary `I1` (the "source zone").
   - Select `SET COUP MODE`.
   - Click on the choice under `CONNECT MODE` (initially `** NONE **`).
   - Respond to the prompts at the bottom of the screen, using `1` (rotation) as the connection mode, a point on the x axis as the center of rotation, and `36.0 0.0 0.0` for the rotation angles.
   - Select `CONNECT`.

   (b) For the next rotation:

   - Select `COUPLE`.
   - Select `SEL OTHER BND`.
   - Pick zone 2, boundary `I1` (the "source zone").
   - Select `SET COUP MODE`.
   - Click on the choice under `CONNECT MODE` (now set to `ROTATION`).
   - Respond to the prompts at the bottom of the screen as before, except using `72.0 0.0 0.0` for the rotation angles.
   - Select `CONNECT`.

   (c) Repeat the previous step for rotation angles of 108°, 144°, 180°, 216°, 252°, 288°, and 324°.

5. Couple the zone 1 `IMAX` surface to the zone 2 `I1` surface in its non-rotated position.

   - Select `BOUNDARY COND`.
   - Pick zone 1, boundary `IMAX` (the "boundary zone").
   - Select `MODIFY BNDY`.
   - Select `COUPLE`.
   - Select `SEL OTHER BND`.
   - Pick zone 2, boundary `I1` (the "source zone").
   - Select `COUPLE`.

6. Repeat steps 4 and 5 twice, coupling the zone 1 `IMAX` surface to the `I1` surfaces in zones 3 and 4.
7. Whew!

The relevant keyword input in the input data (*.dat*) file would be

```
ROTATE 0.0 0.0 0.0 -1680.0 0.0 0.0 ZONE 2:4

REL-ROT-ZONE
   ZONE 1 BOUNDARY IMAX SUBSET I LAST LAST J  1   18 K ALL
   ZONE 2 BOUNDARY I1
   ROTATING-ZONE 2
ENDRRZ
REL-ROT-ZONE
   ZONE 1 BOUNDARY IMAX SUBSET I LAST LAST J 19   34 K ALL
   ZONE 3 BOUNDARY I1
   ROTATING-ZONE 2
ENDRRZ
REL-ROT-ZONE
   ZONE 1 BOUNDARY IMAX SUBSET I LAST LAST J 35 LAST K ALL
   ZONE 4 BOUNDARY I1
   ROTATING-ZONE 2
ENDRRZ
```

See Also: ROTATE

RESTART | START — Begin run in specified zone

```
{START | RESTART} [zone [plane]]
```

This keyword allows the user to specify the zone (and plane, for a marching solution) in which the solution will start (or restart) when running in single-processor mode. In parallel mode, the starting zone number is automatically set to 1.

In both single-processor and parallel modes, if this keyword is not used, WIND checks for the existence of a flow (*.cfl*) file. If one exists, the solution will be restarted using the values in the *.cfl* file as initial conditions. If a *.cfl* file does not exist, WIND will create and initialize one.

zone First zone entered. If not specified, the solution will begin in the zone after the last successfully completed zone. However, if Global Newton iteration is being used, the solution must start in zone 1, and specification of a *zone* value other than 1 is not allowed.

plane *i*-plane to start/restart solution when running in PNS marching mode. If omitted, WIND will automatically start at the next *i*-plane after the last one completed.

RHS — Explicit operator control

```
RHS scheme [order [modifier]] [zone_selector]
```

This keyword allows control of the explicit operator used within each zone. The parameter *scheme* specifies the type of differencing scheme, and must be one of the following:

```
CENTRAL, COAKLEY, ROE, ROE_OVER, VANLEER, HLLE
```

The parameter *order* specifies the type of differencing order, and must be one of the following:

```
FIRST, SECOND, THIRD, FOURTH, FIFTH
```

The *modifier* must be one of:

```
CENTRAL, UPWIND, PHYSICAL, UPWINDBIASED
```

Not all combinations of options are valid.

If scheme is `CENTRAL`, second-order central differencing is used, and any values specified for *order* and *modifier* are ignored.

If *scheme* is `COAKLEY`, Coakley differencing is used and the following options are available for *order*. Any *modifier* that is specified is ignored. Following the usual conventions for displaying keyword syntax, optional keyword parameters are inside square brackets. Thus, `RHS COAKLEY` and `RHS COAKLEY SECOND` both give second-order Coakley differencing, fully upwind.

order	Explicit Operator
`FIRST`	First-order, upwind
`[SECOND]`	Second-order, fully upwind
`THIRD`	Third-order, upwind-biased

If *scheme* is `ROE`, `VANLEER`, or `HLLE`, Roe, Van Leer, or HLLE differencing is used, respectively.

If *scheme* is `ROE_OVER`, an alternative implementation of the Roe scheme from the OVERFLOW code is used. This implementation seems to be faster, and includes an entropy fix that prevents expansion shocks.

The HLLE scheme also includes a built-in entropy fix. Otherwise, the HLLE and Roe schemes give very similar results.

For all of the Roe, Van Leer, and HLLE schemes, the following options are available for *order* and *modifier*. Again, optional parameters are inside square brackets. Thus, `RHS ROE`, `RHS ROE SECOND`, and `RHS ROE SECOND PHYSICAL` all give second-order Roe upwind-biased differencing, modified for stretched grids.

order and *modifier*	Explicit Operator
`FIRST [UPWIND]`	First-order, upwind
`FIRST PHYSICAL`	First-order, upwind, modified for stretched grids
`[SECOND [PHYSICAL]]`	Second-order, upwind-biased, modified for stretched grids
`SECOND CENTRAL`	Second-order, central
`SECOND UPWINDBIASED`	Second-order, upwind-biased
`SECOND UPWIND`	Second-order, fully upwind
`THIRD [UPWINDBIASED]`	Third-order, upwind-biased
`THIRD UPWIND`	Third-order, fully upwind
`FOURTH [UPWINDBIASED]`	Fourth-order, upwind-biased
`FOURTH CENTRAL`	Fourth-order, central
`FIFTH [UPWINDBIASED]`	Fifth-order, upwind-biased

If the `RHS` keyword is not used, the second-order upwind-biased Roe scheme with modifications for stretched grids (i.e., `RHS ROE SECOND PHYSICAL`) is used.

Notes

- Roe zonal coupling requires that Roe, Van Leer, or HLLE differencing be used.
- TVD flux limiting, and the default zonal coupling procedure (high-order Roe), can not be used with some of the higher-order explicit operators. See the `TVD` and `COUPLING` keywords for details.
- Some of the explicit operators can not be used in a spatial marching calculation. See the `MARCHING` keyword for details.
- If a first-order upwind explicit operator modified for stretched grids is used, test option 189 must be set.
- A Roe, Van Leer, or HLLE explicit operator must be used in zones containing holes.
- The `ROE_OVER` and `VANLEER` schemes cannot currently be used for multi-species flows.

See Also: `COUPLING`, `MARCHING`, `TVD`, `HLLE`, `TEST 189`

ROLL — Specify roll about one of the coordinate axes

```
ROLL {X | Y | Z} rat x1 x2
```

The ROLL keyword allows the user to specify roll about one of the coordinate axes.

rat Rotation rate in radians per second

x1, x2 Coordinates of the center of rotation. The coordinates specified by *x1* and *x2* depend on the rotation axis, as shown in the following table:

Rotation axis	*x1*	*x2*
X	y	z
Y	z	x
Z	x	y

Note that in this mode, the calculation is done in inertial space, and the grid is assumed to roll through inertial space at the specified rate. The grid velocity imparts the rotating component. Thus, in order to show the velocity relative to the rotating axes, the roll component of velocity must be subtracted (via the CFPOST post-processing package or another mechanism).

Note also that, unfortunately, unless ROLL is used in a true time-accurate mode which moves the grid each time step (not currently available), as well as giving it velocity, the ROLL capability neglects the rotational potential energy, and so is not correct for high rotation rates.

With this keyword, the velocity vector must be aligned with the coordinate axis. (To get an angle of attack, rotate the grid in GMAN.)

See Also: ROTATE

ROTATE — Perform calculation in a rotating frame of reference

`ROTATE` x_{cntr} y_{cntr} z_{cntr} Ω_x Ω_y Ω_z

The `ROTATE` keyword allows the user to compute the flow in a rotating reference frame. The center of rotation and rotation rates are specified via the arguments:

x_{cntr}, y_{cntr}, z_{cntr}	Center of rotation, in grid units
Ω_x, Ω_y, Ω_z	Rotation rates about the three Cartesian coordinate axes, in radians/sec

Note: A similar capability is given by the `ROLL` keyword, in which the grid is rotated about an axis. However, `ROLL` does the calculation in inertial space, and uses the grid velocity to impart a rotating component. Unfortunately, unless `ROLL` is run in a true time-accurate mode which moves the grid each time step (not currently available), as well as giving it velocity, the `ROLL` capability neglects the rotational potential energy, and so is not correct for high rotation rates.

See Also: `REL-ROT-ZONE`, `ROLL`, `ARBITRARY INFLOW`

SEQUENCE — Grid sequencing control

```
SEQUENCE nsi nsj nsk [zone_selector]
```

This option allows the user to, in effect, thin the computational grid, thus reducing CPU requirements. It can be used to obtain quick starting solutions for the fine grid. Separate keywords may be used for each zone.

nsi Number of sequencing levels in the i direction

nsj Number of sequencing levels in the j direction

nsk Number of sequencing levels in the k direction

For one sequencing level, every other grid point is used in the calculation. For two sequencing levels, the sequencing process is repeated, resulting in only every fourth grid point being used. And so on. At the end of the run, the solution is interpolated back onto the entire grid to aid in restarting the solution and to provide a continuous field for post-processing.

There are no restrictions on the number of points in the sequenced direction, except that there must be at least five points in the sequenced (i.e., coarse) grid.

SMOOTHING — Add dissipation to explicit operator

```
SMOOTHING [SECOND val1] [FOURTH val2] [SMLIMT val3] [zone_selector]
```

This keyword is used to apply second- and/or fourth-order dissipation to the explicit operator. The parameters *val1*, *val2*, and *val3* are the second-order smoothing coefficient, the fourth-order smoothing coefficient, and the smoothing limiter, respectively.

Parameter	Default	"TVD-Like" Value
val1	1/4	< 1
val2	1/256	$< 1/32$
val3	1/4	1

The default values shown in the above table represent a Jameson-type smoothing, and are good for an Euler analysis. For viscous problems, it is recommended that *val1*, *val2*, and *val3* be set equal to 1, 1/64, and 2, respectively.

See Also: BOUNDARY-DAMP, TEST 49

SPAWN — Run an external process from WIND

```
SPAWN "process" [FREQUENCY freq] [NOCHECKPOINT]
```

This keyword allows the user to start (spawn) a process without stopping WIND. The process can be any valid Unix script or executable. When the new process is spawned, WIND will wait until the spawned process completes before continuing.

WIND will tack the current cycle number onto the end of the *process* call for potential use by the process as a parameter.

The spawned process will be run in the same directory as WIND. See the description of the **-runinplace** and **-runroot** options to the *wind* script in Section 8.1 for details about the WIND run directory.

The keyword arguments and options are:

process	Full path name to the process (must be in quotes, 80 characters max)
freq	Spawn process every *freq* cycles (default is 1)
`NOCHECKPOINT`	By default, WIND will override the checkpoint interval so that it aligns with the spawn frequency. `NOCHECKPOINT` will disable the realignment and cause the checkpoint interval to be unchanged.

Note that while the *process* is being executed, WIND's input and output files (i.e., the *.lis* file, *.cfl* file, etc.) remain connected to the Fortran units used by WIND. Thus, if the *process* runs a program like *resplt* or *cfpost*, these files cannot be used directly as input. Instead, the *process* must copy the needed file(s) to temporary files, and use the temporary files as input.

Example

A common use for the `SPAWN` keyword, especially for unsteady flows, is to save intermediate *.cfl* files during a WIND run. A shell script may be `SPAWN`'ed at specified intervals to save the file under a unique name.

For example, consider the following shell script, named *save_cfl*.

```
#! /bin/sh
#
# save_cfl - Save intermediate WIND .cfl files.  This script is intended
#            to be spawned during a WIND run using the SPAWN keyword, as
#
#               SPAWN "path-to-script path-to-cfl" FREQUENCY num
#
#            where 'path-to-script' is the full path name for this
#            script, 'path-to-cfl' is the full path name for the saved
#            .cfl files (the current cycle number will automatically
#            be appended to this name), and 'num' is the frequency in
#            cycles for the SPAWN command.  Note that the quote marks
#            around "path-to-script path-to-cfl" must be included.
```

```
# Check for enough arguments

if [ $# -lt 2 ]
then
   echo 'Error: save_cfl: Path name for saved .cfl files not specified' >& 2
   echo '       Use WIND''s SPAWN keyword, as:' >& 2
   echo '       SPAWN "path-to-script path-to-cfl" FREQUENCY num' >& 2
   exit 1
fi

# $1 is path name for the saved .cfl files, $2 is current cycle number

cp FOR020 $1.$2
```

If the above script is located in the directory */home/user/bin*, the `SPAWN` keyword may be used as follows to save the *.cfl* file every 1000 cycles in the directory */home/user/wind*.

```
SPAWN "/home/user/bin/save_cfl /home/user/wind/runa.cfl" FREQUENCY 1000
```

Since the cycle number is automatically added to the end of the *process* when it's spawned, the above use of the `SPAWN` keyword causes the *save_cfl* script to be run as

```
/home/user/bin/save_cfl /home/user/wind/runa.cfl ncycle
```

where *ncycle* is the current cycle number. The result is that the intermediate *.cfl* files will be stored in the directory */home/user/wind*, with names *runa.cfl.1000*, *runa.cfl.2000*, etc.

STAGES — Multi-stage time stepping

```
STAGES ityp [COEFFICIENTS x1 x2 ... xn] [zone_selector]
```

This keyword allows the user to control the Runge-Kutta time stepping option. The parameter *ityp* specifies the number of stages, and *x1 x2 ... xn* specify the coefficients. The following *ityp* values are available:

ityp	Stages
0	One stage, Euler implicit or explicit
1	Four stage, Jameson type
5	Three stage, minimal storage, third order

The default coefficients are:

ityp	# Req'd	Defaults
0	—	—
1	4	1/4, 1/3, 1/2, 1
5	5	1/4, 8/15, 0, 5/12, 3/4

TDA_INVALID — Rewrite boundary data to *.tda* file

```
TDA_INVALID [ZONE] range1[,range2[, ...  rangen]]
```

This keyword tells WIND that the boundary data from the *.tda* file for the specified zone(s) should not be used, until new boundary data is written to the file during the first solution cycle. In the zone specification, the *range* parameter(s) must be one of the following forms:

zonenum	Selects zone *zonenum*
begzone:*endzone*	Selects all zones from *begzone* to *endzone*
`ALL`	Selects all zones

TEST — Non-production test options

```
TEST number [mode]
```

For purposes of program testing, the user may select one or more test options for any job. These options are selected by the keyword `TEST` (beginning in column 1) followed by one or two integer variables. The keyword and the integer variables must be separated by one or more blanks. The first integer selects the test option. The second integer, which is not mandatory, selects a mode for that option. If the second integer is omitted, a default mode is activated. Any number of valid test options may be selected. However, only one mode may be activated for each option. If an option is selected more than once, the mode identified in the last selection will be used.

The list of valid options and modes is provided in Section 11.

TSL | THIN SHEAR LAYER — Thin shear layer option

```
{TSL | THIN} [SHEAR] [LAYER] word1 word2 word3 [zone_selector]
```

This keyword turns off the viscous terms in specified coordinate directions. The parameters *word1*, *word2*, and *word3* are keywords controlling the viscous terms in the ξ-, η-, and ζ-directions, respectively. They may have the following values:

Keyword	Meaning
VISCOUS	Retain all viscous terms in this direction
INVISCID	Neglect all viscous terms in this direction

By default, when INVISCID is set for a direction, the scalar implicit operator will be used in that direction (the full block implicit operator is used in viscous directions). This may be overridden using the IMPLICIT keyword.

TTSPEC — Wall temperature and transition (block)

```
TTSPEC
   TYPE TEMPERATURE | TRANSITION
   MODE FILE filename | CURRENT
   ZONE zone_number
   SURFACE surfname
   SUBSET ALL
ENDTTSPEC
```

This keyword block is used when a point-by-point surface temperature boundary condition is being used, and/or when point-by-point boundary layer transition information is being specified. Surface temperature and transition data may be specified for any boundary surface in any zone.[27] The maximum total number of temperature/transition surfaces, combined, is determined by the Fortran parameter `MAXTTS`, currently equal to 20.

The normal procedure is to do an initial run, perhaps with just a few cycles, without specifying temperature/transition information, to create the initial common flow (*.cfl*) file. The temperature/transition data is then created and added to the *.cfl* file using the *tmptrn* utility.

For backward compatibility with WIND 2.0, however, the temperature/transition data may be read from separate ASCII files, one for each applicable surface, created with the older *tmptrn* utility. This would only be necessary during the initial (i.e., non-restart) run, since WIND will automatically store the data in the *.cfl* file, where it can be read during subsequent runs.

In addition, in earlier versions of *tmptrn*, the temperature written into the *.cfl* file was in K, instead of being non-dimensionalized. This error was fixed in version 1.8 of *tmptrn*. However, some versions of Wind-US had coding to accomodate the dimensional temperature value, and these versions will not work with temperature distributions written using *tmptrn* 1.8 and above. See the *tmptrn* documentation for details.

The various elements of the `TTSPEC` keyword block are defined as follows:

```
TYPE TEMPERATURE | TRANSITION
```

This keyword indicates whether a surface temperature distribution or boundary layer transition information is being specified.

```
MODE FILE filename | CURRENT
```

This keyword specifies where the surface temperature or transition data is stored.

`FILE` *filename* — The data is stored in the ASCII file *filename*, created using the older *tmptrn* utility distributed with WIND 2.0. If *filename* is specified without an extension, an extension of *.dat* is assumed.

This option is included for backward compatibility only, using existing files created with the older *tmptrn* utility distributed with WIND 2.0, and is only

[27] Note, though, that although the WIND code allows temperature/transition data to be specified on any boundary surface, the *tmptrn* utility used to add the data to the *.cfl* file currently does not allow specification on the i_1 or i_{max} surface.

needed for an initial (i.e., non-restart) run. Subsequent (i.e., restart) runs should use `MODE CURRENT`. New applications should use the latest *tmptrn* utility to write the temperature/transition data into the *.cfl* file directly.

`CURRENT` The data is stored in the flow (*.cfl*) file.

`ZONE` *zone_number*

This keyword specifies the number of the zone containing the boundary surface along which temperature or transition data is being specified.

`SURFACE` *surfname*

This keyword specifies the boundary surface in zone *zone_number* along which temperature or transition data is being specified. Currently, the applicable surface names are `I1`, `IMAX`, `J1`, `JMAX`, `K1` and `KMAX`, corresponding to the i_1, i_{max}, j_1, j_{max}, k_1 and k_{max} surfaces, respectively.

`SUBSET ALL`

In the future, this keyword will allow temperature/transition information to be specified over a subset of the surface. Currently, `SUBSET ALL` is only used to update the counter of the number of temperature/transition surfaces, and must be specified once for each surface.

Example

As an example, suppose we are specifying the wall temperature distribution along the j_1 surface in zone 1, and the j_1 and j_{max} surfaces in zone 2. In addition, suppose we're specifying boundary layer transition along the j_1 surface in zone 1. If existing temperature/transition files are being used, created with the older *tmptrn* utility, the `TTSPEC` keyword block for the initial run might be

```
TTSPEC
   TYPE TEMPERATURE
      ZONE 1
         MODE FILE temp.z1j1.asc
         SURFACE J1
         SUBSET ALL
      ZONE 2
         MODE FILE temp.z2j1.asc
         SURFACE J1
         SUBSET ALL
         MODE FILE temp.z2jm.asc
         SURFACE JMAX
         SUBSET ALL
   TYPE TRANSITION
      ZONE 1
         MODE FILE tran.z1j1.asc
         SURFACE J1
         SUBSET ALL
ENDTTSPEC
```

Since the temperature/transition data will be written into the *.cfl* file during the initial run, later (i.e., restart) runs could use

```
TTSPEC
   TYPE TEMPERATURE
      ZONE 1
         MODE CURRENT
         SURFACE J1
         SUBSET ALL
      ZONE 2
         MODE CURRENT
         SURFACE J1
         SUBSET ALL
         MODE CURRENT
         SURFACE JMAX
         SUBSET ALL
   TYPE TRANSITION
      ZONE 1
         MODE CURRENT
         SURFACE J1
         SUBSET ALL
ENDTTSPEC
```

See Also: WALL TEMPERATURE

TURBULENCE — Turbulence model selection

```
TURBULENCE [MODEL] model [RC | ROTATION] [F2FIX dmax] [ITERATIONS iter]
   [zone_selector]
```

This keyword is used to request an inviscid or viscous solution, and to select a turbulence model for one or more zones. The parameter *model* must be one of the following keywords:

`INVISCID | EULER`	Euler solution (scalar implicit)
`LAMINAR`	Viscous, laminar
`CEBECI [SMITH]`	Viscous, Cebeci-Smith algebraic model
`[BALDWIN] LOMAX`	Viscous, Baldwin-Lomax algebraic model
`PDT`	Viscous, combination Baldwin-Lomax and P. D. Thomas algebraic shear layer model
`[BALDWIN] BARTH`	Viscous, Baldwin-Barth one-equation model
`SPALART [ALLMARAS]`	Viscous, Spalart-Allmaras one-equation model. Additional keywords may be used with the Spalart-Allmaras model to specify the freestream turbulence level (p. 189), and to specify use of the Spalart Detached Eddy Simulation (DES) turbulence model (p. 190). In multi-zone problems, the Spalart-Allmaras model may not be used with different turbulence models in other zones, other than the Spalart DES model. However, it may be used with inviscid or laminar flow in other zones.
`SST`	Viscous, two-equation Shear Stress Transport model. Additional keywords specific to the SST model are available, described starting on p. 190, for specifying freestream values of k and ω. Another additional keyword may be used to specify use of a combined SST and Large Eddy Simulation (LES) turbulence model, as described starting on p. 191. In multi-zone problems, the SST model may not be used with different turbulence models in other zones, other than the combined SST and LES model. However, it may be used with inviscid or laminar flow in other zones.
`CHIEN [K-E]`	Viscous, Chien low-Reynolds-number two-equation model. Additional keywords specific to the Chien model are also available, described starting on p. 192. When the Chien model is used for multi-zone problems, it must be used in all the zones.

Note that, with the exceptions of the Spalart, SST, and Chien models as noted above, different turbulence models, or inviscid or laminar flow, may be specified in different zones. *However, you must specify a "default" turbulence model (or inviscid or laminar flow) in the input data file. WIND will stop if you do not.* By "default", we mean without specifying zones.

For example, for a three-zone problem with inviscid flow in zone 1 and the Spalart-Allmaras turbulence model in zones 2 and 3, the following will *not* work, and the code will stop:

```
TURBULENCE INVISCID ZONE 1
TURBULENCE SPALART ZONE 2,3
```

Instead, one could specify the following, which will work:

```
TURBULENCE INVISCID
TURBULENCE SPALART ZONE 2,3
```

Additional parameters that may be used with the TURBULENCE keyword are:

RC | ROTATION	May be used with the Spalart-Allmaras one-equation model to include a correction for system rotation and streamline curvature
F2FIX *dmax*	May be used with the SST two-equation model to specify a maximum distance from the wall, *dmax*, within which the F_2 term may be non-zero. Beyond that distance, the F_2 term is set to zero. F_2 is a blending function, designed to limit the effects of the shear stress transport term to regions near walls.
ITERATIONS *iter*	The WIND code organizes the equations to be solved into logical "groups" that are solved together. It also allows multiple iterations of a specific group (i.e., sub-iterations) for each "iteration per cycle". (The number of iterations per cycle is set using the ITERATIONS keyword.) For one- and two-equation turbulence models, this parameter allows the user to request *iter* sub-iterations of the turbulence model equation group for each "iteration per cycle". If NAVIER-STOKES ITERATIONS is defaulted, this corresponds to *iter* iterations of the turbulence model equations for each iteration of the mean flow equations. The default for *iter* is one, indicating that each "iteration per cycle" corresponds to one iteration of the turbulence model equations.

Note that for the one- and two-equation turbulence models, the turbulence model equations may be solved without simultaneously solving the Navier-Stokes equations. Of course, the turbulence variables depend on the mean flow field, so a reasonable mean flow solution must already exist.

As an example, suppose a mean flow solution has been computed using the SST turbulence model. The Chien k-ϵ variables consistent with the existing mean flow field may be computed by restarting from the SST solution, and solving just the Chien k-ϵ equations.

```
ITERATIONS PER CYCLE 2 ZONE ALL
NAVIER-STOKES ITERATIONS 0 ZONE ALL
TURBULENCE MODEL CHIEN ITER 5 ZONE ALL
```

The above keywords specify that, for all zones, there will be two iterations per cycle, with no Navier-Stokes sub-iterations and five Chien model sub-iterations for each "iteration per cycle". There will thus be a total of ten iterations of the Chien k-ϵ equations in each zone prior to completing a cycle and exchanging information between zones.

See Also: ITERATIONS, NAVIER-STOKES ITERATIONS, TEST 21, TEST 67

Spalart-Allmaras Model

FREE_ANUT *anutinf*

The separate keyword FREE_ANUT may be used with the Spalart-Allmaras model to specify the freestream value of the eddy viscosity ν_t. The default value is 5.0.

See Also: TEST 21

Spalart DES Model

```
DES [CDES cdes] [zone_selector]
```

The separate keyword DES may be used with the Spalart-Allmaras model to specify use of the Spalart Detached Eddy Simulation (DES) turbulence model. It is intended to improve the results for unsteady and massively separated flows. The DES model reduces to the standard Spalart-Allmaras model near viscous walls, where the grid is refined and has a large aspect ratio, but acts like a Large Eddy Simulation (LES) model away from the boundary, where the grid is coarser and has an aspect ratio of order one.

The input parameter *cdes* specifies the value of the coefficient C_{des} in the model. The default value is 0.65. Increasing C_{des} increases the size of the region in which the DES model reduces to the standard Spalart-Allmaras model. Details may be found in papers by Spalart, Jou, Strelets, and Allmaras (1997) and by Shur, Spalart, Strelets, and Travin (1999).

The DES model may only be used for unsteady (i.e., the time step is a constant) three-dimensional flows. It is zonal, however, so you can use the DES model in time-accurate mode in one zone, while using some other model in steady-state mode in the other zones. For example, a three-zone problem could use the regular Spalart-Allmaras model with a specified CFL number in zones 1 and 2, and the DES model with a specified time step in zone 3, using the following keywords:

```
TURBULENCE SPALART
DES ZONE 3
CFL# 1.3
TIMESTEP SECONDS 5.0E-6 ZONE 3
```

This capability can accelerate the solution convergence tremendously, especially for large configurations (10 to 20 million grid points) that would be impossible to run in time-accurate mode throughout the flow field.

SST Model

```
FREE_K val_k
FREE_OM val_om
```

The separate keywords FREE_K and FREE_OM may be used with the SST model to specify dimensional freestream values of k and ω. The following options are possible:

val_k > 0 The turbulent kinetic energy k and the specific dissipation rate ω are specified directly, with

$$k = val_k \ (\mathrm{ft}^2/\mathrm{sec}^2)$$
$$\omega = val_om \ (1/\mathrm{sec})$$

The turbulent viscosity ν_t is then equal to k/ω.

val_k < 0 The turbulent kinetic energy k is set equal to *val_k* percent of the "reference" kinetic energy $U_\infty^2/2$, where U_∞ is the freestream velocity. Thus

$$k = 0.01\,|val_k|\,\frac{U_\infty^2}{2}$$

The turbulent viscosity ν_t is automatically set equal to $0.001\nu_l$, where ν_l is the laminar viscosity, and the specific dissipation rate is computed as $\omega = k/\nu_t$.

val_om < 0 The specific dissipation rate ω is set equal to *val_om* percent of U_∞/L_{ref}, where U_∞ is the freestream velocity, and L_{ref} is the reference length from the grid (*.cgd*) file. Thus

$$\omega = 0.01\,|val_om|\,\frac{U_\infty}{L_{ref}}$$

The turbulent viscosity ν_t is set to the same percentage of the laminar viscosity.

$$\nu_t = 0.01\,|val_om|\,\nu_l$$

The turbulent kinetic energy is then computed as $k = \omega\nu_t$.

The default, and recommended, values are

$$\omega = 10U_\infty/L$$
$$k = \omega\nu_t$$

where the freestream turbulent viscosity ν_t is arbitrarily set to $0.001\nu_l$.

Note that

- If *val_k* > 0, a positive value must be specified for *val_om*.
- If *val_k* ≤ 0, *val_om* should not be specified.
- If *val_om* < 0, any value specified for *val_k* is ignored.

Inflow turbulence levels may be specified for the SST model in the ARBITRARY INFLOW keyword block. If this is done, the TURBULENCE keyword must come before the ARBITRARY INFLOW keyword block in the input data (*.dat*) file.

See Also: ARBITRARY INFLOW

Combined SST and LES Model

LESB [CBLES *cb*] [*zone_selector*]

The separate keyword LESB may be used with the SST model to specify use of a combined SST and Large Eddy Simulation (LES) turbulence model. The intent is to improve predictions of complex flows in a real-world engineering environment, by allowing the use of LES methods with grids typical of those used with traditional Reynolds Averaged Navier Stokes models.

The combined model reduces to the standard SST model in high mean shear regions (e.g., near viscous walls), where the grid is refined and has a large aspect ratio unsuitable for LES models.

As the grid is traversed away from high mean shear regions, it typically becomes coarser and more isotropic, and the combined model smoothly transitions to an LES model.

The input parameter *cb* specifies the value of the coefficient C_B in the model. The default value is 10.0. Increasing C_B increases the size of the region in which the combined model reduces to the standard SST model.

The combined model may only be used for unsteady flows (i.e., the time step is a constant). It is zonal, however, so you can use the combined model in time-accurate mode in one zone, while using the standard SST model in steady-state mode in the other zones. For example, a three-zone problem could use the standard SST model with a specified CFL number in zones 1 and 2, and the combined model with a specified time step in zone 3, using the following keywords:

```
TURBULENCE SST
LESB ZONE 3
CFL# 1.3
TIMESTEP SECONDS 5.0E-6 ZONE 3
```

Chien k-ϵ Model[28]

Several keywords may be used with the Chien k-ϵ model to control the initialization procedure, enhance its stability, and improve its accuracy in adverse pressure gradients and at high Mach numbers. For convenience, all keyword phrases associated with the Chien k-ϵ model begin with `K-E`.

```
K-E INITIALIZE [FROM] {EXISTING | EQUILIBRIUM | FREESTREAM}
```

This keyword determines how the turbulent transport variables for the Chien k-ϵ model (k, ϵ, μ_t) will be initialized.

The first method, given by the `EXISTING` parameter, performs an "intelligent" initialization, based on the type of turbulence model (if any) used in the previous run. This is the default.

The second method, given by the `EQUILIBRIUM` parameter, uses an assumption of turbulent equilibrium, namely that the production, Π, of turbulent kinetic energy equals the rate of dissipation, together with an existing turbulent viscosity profile to initialize the k and ϵ variables.

$$\rho\epsilon = \Pi/Re$$
$$\rho k = \sqrt{\frac{\rho\epsilon\mu_t}{C_\mu f_\mu Re}}$$

In order to use this technique, the user must first run a few thousand iterations using another eddy viscosity turbulence model. Initializing from an existing turbulent viscosity profile rather than uniform values aids somewhat in convergence and improves the stability of the model by reducing the dramatic changes in turbulence values that occur during the first few iterations after initialization.

The third method, given by the `FREESTREAM` parameter, initializes the turbulence variables to uniform values within each zone.

$$\rho k = \rho k_\infty$$
$$\rho\epsilon = \rho\epsilon_\infty$$
$$\mu_t = (\mu_t)_\infty$$

[28]The material in this section was originally written by Dennis Yoder of NASA Glenn Research Center.

where the local density is used and the freestream conditions k_∞ and $(\mu_t)_\infty$ may be specified with the keywords K-E FREE_K and K-E FREE_MUT.

`K-E FREE_K` *val1*
`K-E FREE_MUT` *val2*

These keywords may be used to specify the freestream turbulence values to be used when initializing the turbulence variables to uniform values (i.e., with the `K-E INITIALIZE FREESTREAM` keyword). The interpretation of the input depends on the signs of *val1* and *val2*.

For the turbulence kinetic energy k_∞,

$$k_\infty = \begin{cases} \frac{3}{2} I^2 u_\infty^2 & \text{for } val1 < 0, \text{ and where } |val1| = I \\ \bar{k}_\infty / \bar{a}_\infty^2 & \text{for } val1 > 0, \text{ and where } val1 = \bar{k}_\infty \text{ (ft/sec)}^2 \end{cases}$$

and for the turbulent viscosity $(\mu_t)_\infty$,

$$(\mu_t)_\infty = \begin{cases} (\mu_t)_\infty & \text{for } val2 < 0, \text{ and where } |val2| = (\mu_t)_\infty \\ (\bar{\mu}_t)_\infty / \bar{\mu}_\infty & \text{for } val2 > 0, \text{ and where } val2 = (\bar{\mu}_t)_\infty \text{ (slug/ft-sec)} \end{cases}$$

where the overbar denotes a dimensional value.

If the `K-E FREE_K` keyword is not used, or if $val1 = 0$, then a default value of $val1 = -0.01$ is used, which corresponds to a turbulence intensity I of 1%. If the `K-E FREE_MUT` keyword is not used, or if $val2 = 0$, then a default value of $val2 = -0.001$ is used. This sets the freestream turbulent viscosity to be 0.001 times the freestream laminar viscosity. Note that for values greater than zero, the expected units for *val1* and *val2* are $(\text{ft/s})^2$ and slug/ft-s, respectively.

These freestream values are only used when the k-ϵ model is initialized from uniform conditions. Currently, the model extrapolates to get values at all inflow and outflow boundaries. Thus, `K-E FREE_K` and `K-E FREE_MUT` can not be used to specify inflow conditions.

`K-E REINITIALIZE`

Under certain conditions, the k-ϵ information in the flow (*.cfl*) file can become "contaminated" and the model must be forced to reinitialize. Consider the following example. An attempt to initialize the k-ϵ model too soon results in poor k and ϵ profiles. So, the user decides to continue using another turbulence model until the flow is more well defined. When he later tries to reinitialize the k-ϵ model, he finds he has the same problem. This occurs because the *.cfl* file does not remove outdated information. Thus, when he specifies the k-ϵ model, the code reads the old k and ϵ information from the previous run.

Using the keyword phrase `K-E REINITIALIZE` signals the code to ignore the old k-ϵ information in the *.cfl* file and perform a fresh initialization. This command must be removed on subsequent runs or else the model will reinitialize itself each time.

`K-E [TVD] ORDER {1|2|3}`

This keyword sets the spatial order of accuracy of the TVD upwinding used in solving the k-ϵ equations. The default is first order.

`K-E RELAX [FOR]` *val* `[ITERATIONS]`

Updated values of k, ϵ, and μ_t will be relaxed for *val* iterations (the default is 500) following the initialization. Relaxation of each of these variables reduces the amount they may change during any

single iteration. Immediately after initialization, the allowed changes are significantly reduced. This restriction is then gradually lifted as the last relaxation iteration is approached.

```
K-E [MAXIMUM] [TURBULENT] VISCOSITY val1
K-E [TURBULENT] [REFERENCE] VELOCITY val2
```

The k-ϵ model uses limiters within the interior of each zone to increase convergence and stability by capping the values of the turbulence quantities at both the high and low extremes. This is usually only necessary during the first few iterations after initialization, when the fluctuations in k and ϵ tend to be the most dramatic.

Nondimensional values of the minimum limiters have been preset to small numbers. The minimum turbulent viscosity is set to 0.001, k_{min} is set to 10^{-9}, and ϵ_{min} is computed from the turbulent viscosity relation using an assumed reference density of 1, and $f_\mu = 1$.

The above keywords determine the maximum limiting values for the turbulence quantities. The first keyword sets the maximum turbulent viscosity to be *val1* (the default is 10,000) times the freestream viscosity. The second keyword sets the turbulent reference velocity equal to *val2* (the default is 1.0) times the freestream speed of sound. The maximum turbulent kinetic energy allowed is 10% of the kinetic energy of the turbulent reference velocity:

$$k_{max} = 0.10 \frac{u^2_{ref(k\text{-}\epsilon)}}{2}$$

The maximum dissipation rate is again computed from the turbulent viscosity relation.

The use of these limiters can be summarized as follows:

- If either k or ϵ fall below the preset minimum values then both are reset to these values. This typically occurs in the freestream.
- If the turbulent kinetic energy exceeds k_{max}, then it is capped at this value. The dissipation rate is taken to be the larger of the current dissipation rate or ϵ_{max}.
- If the turbulent viscosity exceeds $(\mu_t)_{max}$, then it is capped at this value and the turbulent kinetic energy is recomputed from the turbulent viscosity relation. The turbulent dissipation rate is left unchanged.

If the maximum limiters cause the turbulence variables to be capped within the flowfield, a warning message will be written to the list output (*.lis*) file. By using CFPOST to examine the normalized variable `mut muinf`, one can observe where these limiters are being used and adjust them using the above keywords. It is important that the turbulence values not be limited upon convergence.

```
K-E [VARIABLE] CMU {ON|OFF}
```

It is well known that the baseline k-ϵ model is poorly suited to adverse pressure gradient flows, such as those found in diffusers. Rodi and Scheuerer (1986) demonstrated that for these types of flows, the rate of dissipation near solid boundaries is too small relative to the rate of production of turbulent kinetic energy. This causes the model to overpredict skin friction and predict flows to be attached when experimental results show them to be separated.

The variable C_μ formulation, which is derived from algebraic stress modeling, is designed to help remedy this problem by reducing the turbulent viscosity in regions of the flowfield where the production of turbulent kinetic energy is significantly larger than the rate of dissipation. The specific formulation used is:

$$C_\mu = \min\left(0.09, \frac{0.10738(0.64286 + 0.19607R)}{[1 + 0.357(R-1)]^2}\right)$$

As the ratio R of production to dissipation increases above 1, the coefficient C_μ is reduced from its normal value of 0.09 to limit the turbulent viscosity.

The variable C_μ option also provides added stability to the k-ϵ model, such as in the case of an airfoil, where the sudden deceleration of the flow near the leading edge would otherwise result in a significant rate of production. In regions of the flow where the turbulence is in equilibrium, i.e., where the production and dissipation are balanced, the turbulent viscosity remains unchanged.

The above keyword may be used to turn this option on or off (the default is `ON`).

`K-E COMPRESSIBILITY [CORRECTION] {NONE | OFF | SARKAR | WILCOX | USER` α_k M_{t_0}`}`

This keyword may be used to specify a compressibility correction, designed to enhance predictions at higher Mach numbers. The equation for the compressibility correction is

$$F(M_t) = \alpha_k \max\left(M_t^2 - M_{t_0}^2, 0\right)$$

The turbulent Mach number M_t is defined from $M_t^2 = 2k/a^2$, where a is the local speed of sound.

The parameters α_k and M_{t_0} for the various options are defined in the following table.

Correction Type	α_k	M_{t_0}
`NONE | OFF`	0.0	0.00
`SARKAR`	1.0	0.00
`WILCOX`	1.5	0.25

Note that both `NONE` and `OFF` disable the compressibility correction, and that values of α_k and M_{t_0} should only be included when using the `USER` option. The default is `SARKAR`.

Both the Sarkar and Wilcox compressibility corrections are designed to improve the prediction of compressible jet flows by including the compressible portion of the dissipation rate in the transport equation for the turbulent kinetic energy. These corrections use simple algebraic relations between the solenoidal and compressible dissipation rates. The effect of these corrections is to reduce the turbulent kinetic energy in regions where the flow is supersonic. In terms of supersonic jet predictions, this results in slower spreading rates, reduced mixing, and a longer core length.

See Also: `TEST 29`, `TEST 51`

Using CFPOST

Care must be taken when using the CFPOST post-processing program to examine turbulence variables. Since the common flow (*.cfl*) file does not remove outdated information, it may contain turbulence variables for several different turbulence models.

For example, suppose an initial case was run using the SST model, then restarted using the k-ϵ model. The *.cfl* file will contain values for the turbulent quantities k, ω, ρk, $\rho\epsilon$, and μ_t. To examine the k-ϵ variables ($\rho k, \rho\epsilon$), with the CFPOST `variables` command the user should specify the variables `rho*k` and `rho*epsi`. Specifying just `k` in CFPOST will give the (older) turbulent kinetic energy from the SST model, not from the k-ϵ model.

The freestream parameters `kinf` and `einf` in the *.cfl* file will be those for whichever model was run most recently. They should therefore not be used to normalize turbulence quantities from earlier runs that used a different turbulence model.

In addition, since all the models use a turbulent viscosity, they share the variable μ_t, which is specified in CFPOST as `mut`. The values displayed for μ_t will be for whichever eddy viscosity model (even the Baldwin-Lomax model) was used most recently.

TVD — Total Variation Diminishing operator flag

```
TVD [OFF | FACTOR factor | MINMOD [factor] | KOREN [factor] | ALBADA] \
    [zone_selector]
```

This keyword controls the TVD flux limiter in the explicit operator, which is helpful in preventing overshoots in flowfield properties in regions of high gradients. TVD may be used with the Coakley, Roe, Van Leer, and HLLE explicit operators. TVD is enabled by default for these operators, using the standard minmod TVD limiter with the default compression parameter, and disabled for all others.

`OFF` Disables TVD for the specified zone(s)

`FACTOR` Use the standard minmod TVD limiter. The value of *factor* specifies the "compression" parameter for the TVD algorithm, which controls the amount of limiting applied.

`FACTOR` Use the standard minmod TVD limiter. This is the same as `FACTOR`, except that the *factor* may be defaulted. The value of *factor* specifies the "compression" parameter for the TVD algorithm, which controls the amount of limiting applied. The default compression parameter is the maximum value which assures TVD properties for the explicit operator being used (as specified by the `RHS` keyword).

For the Coakley explicit operator, the default compression parameter is 0 (i.e., not applicable) for first-order upwind, and 2 for both second-order fully upwind and third-order upwind-biased differencing.

For the Roe, Van Leer, and HLLE explicit operators, the default compression parameter is shown in the following table for the various versions of the operator.

Explicit Operator	*factor*
First-order, upwind	0 (i.e., not applicable)
First-order, upwind, modified for stretched grids	2
Second-order fully upwind	2
Second-order central	2
Second-order upwind-biased	3
Second-order upwind-biased, modified for stretched grids	3
Third-order upwind-biased	4
Third-order fully upwind	Not available
Fourth- and fifth-order	Not available

`KOREN` Use the Koren TVD limiter. With this limiter, the default value of *factor* is 1.0, and does not depend on the explicit operator being used.

`ALBADA` Use the van Albada TVD limiter. With this limiter, no *factor* value is input.

If TVD flux limiting is turned on for an invalid explicit operator, including the "not available" Roe, Van Leer, and HLLE operators, an error message is generated and the run will abort.

See Also: `BOUNDARY TVD`, `RHS`

VORTEX GENERATOR — Vortex generator model (block)

```
VORTEX [GENERATOR]
   ZONE iz1 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
        [SUBSET I range J range K range]
   ZONE iz2 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
        [SUBSET I range J range K range]
   NUMBER ival
     vg_boundary {XLOC xl | YLOC yl | ZLOC zl} chord height alpha \
        [VEL vmax] [DEL delta] ENDVORTEX
```

This keyword block is applicable to three-dimensional cases and enables the user to specify a discontinuous change in secondary velocity across a zone boundary in order to simulate the vortices produced by an array of vane-type vortex generators. A vortex generator array consists of one or more vortex generators mounted on viscous wall boundaries. A separate vortex generator keyword block must be used for each array. The model is only applicable to generators placed on viscous wall boundaries. It is the user's responsibility to check this — WIND currently has no mechanism for flagging this input error.

The vortex generator model determines the strength of each vortex based on the generator chord length, height and angle of incidence with the incoming flow, as well as the incoming flow core velocity and boundary layer thickness. Each vortex center is placed at the grid point closest to the location determined by the user-specified generator location on the boundary, and the generator height.

Note that the vortex generator model is derived from experimental data taken at axial stations which were one chord length downstream of the trailing edge of the generators, and therefore the secondary velocities produced by the model simulate vortices at this station, rather than at the generator trailing edges. For most accurate results, the boundary where the vortex generator boundary condition is applied should be at this one chordlength station.

The following restriction applies:

- The `BOUNDARY TVD FACTOR 0` keyword option should be used at all interface boundaries containing vortex generators.

The various elements of the `VORTEX GENERATOR` input block are defined as follows:

```
VORTEX [GENERATOR]
```

Defines the beginning of the vortex generator keyword input block.

```
ZONE iz1 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
     [SUBSET I range J range K range]
ZONE iz2 BOUNDARY {I1 | IMAX | J1 | JMAX | K1 | KMAX} \
     [SUBSET I range J range K range]
```

These two lines define the location of the vortex generator array. The relevant zones are given by the values of *iz1* and *iz2*, and the relevant boundaries within zones *iz1* and *iz2* are specified via the `BOUNDARY` keyword parameter.

iz1 The "upstream" zone. The secondary velocities will be increased when passing information from this zone to *iz2*.

iz2 The "downstream" zone, i.e., the zone receiving increased secondary velocities from zone *iz1*. Secondary velocities will be decreased when passing information from this zone back to zone *iz1*.

Note: Currently, the vortex generators must be located at either an i_1 or i_{max} boundary. Thus, the only valid choices with the `BOUNDARY` keyword parameter are `I1` and `IMAX`.

Note that the secondary velocity is the flow in a plane normal to the primary velocity. For example, for a vortex generator array at an i-interface boundary, the primary flow is in the i direction, and the secondary velocity is in the j and k directions.

The `SUBSET` parameter may be used to specify that the change in secondary velocity occurs only over a part of the zone boundary. Otherwise, it is assumed that the change occurs over the entire boundary. The *range* parameters define the part of the zone boundary over which the change occurs, and take one of the following forms:

index1 index2 Starting and ending indices in the specified direction. `LAST` may be used for the last index.

`ALL` Equivalent to `1 LAST`.

The starting and ending indices for the appropriate `I`, `J`, or `K` parameter (depending on the boundary specified) must be the same, and correspond to that boundary.

`NUMBER` *ival*

This defines the number of vortex generators in the vortex generator array (i.e., on the specified boundary).

vg_boundary {`XLOC` *xl* | `YLOC` *yl* | `ZLOC` *zl*} *chord height alpha* \
  [`VEL` *vmax*] [`DEL` *delta*]

Defines the location and geometric parameters of each vortex generator. A separate line of input *must* be included for each generator.

vg_boundary The minimum or maximum index, specified as `IM`, `IX`, `JM`, `JX`, `KM`, or `KX`, within the specified boundary surface (as specified by `ZONE` and `SUBSET`) where the generator is mounted. This boundary must have been defined as a viscous wall in GMAN. As an example, for a vortex generator array at an i-interface boundary, either `JM`, `JX`, `KM`, or `KX` should be specified, where `JM` means the generator is at the $j = 1$ interface boundary, `JX` means it's at the $j = j_{max}$ boundary, etc.

`XLOC, YLOC, ZLOC` *xl*, *yl*, or *zl* is the x, y, or z coordinate location in inches of the base of the vortex generator on the already specified boundary surface and wall boundary. (Currently, only one coordinate direction and location may be specified. This may lead to ambiguity in specifying the locations for generators placed in complex duct geometries. This shortcoming will be addressed in future code updates.)

chord	The chord length of the vortex generator in inches. Must be greater than zero.
height	The height of the vortex generator in inches. Must be greater than zero.
alpha	The angle of incidence of the generator in degrees. Must be non-zero. More specifically, it is defined as the angle the generator chord line makes with the primary flow direction. The sign of *alpha* is determined using the "right-hand rule" of vector mechanics. For a given generator, use a normal vector pointing into the wall (the thumb), and the primary flow direction (fingers). If the rotation from the primary flow direction towards the generator follows the right-hand rule, then the sign of *alpha* is positive. Otherwise, it is negative. A positive value of *alpha* generally results in a vortex with a counterclockwise rotation, and conversely. The recommended range of magnitudes for *alpha* is between 8 and 20 degrees.
`VEL`	*vmax* is the maximum velocity at the vortex generator station, in ft/sec. The default is for WIND to compute this value from the flow. It is currently recommended that this value be specified by the user. Must be greater than zero.
`DEL`	*delta* is the boundary layer thickness of the incoming flow, in inches. The default is for WIND to compute this value from the flow. It is currently recommended that this value be specified by the user. Must be greater than zero.

In selecting values for the above described parameters, keep in mind that the vortex generator model was derived from conservation of momentum and inviscid theory, correlated with experimental data having *height*/*chord* ratios between 0.13 and 2.62, *height*/*delta* ratios between 0.12 and 2.60, and for duct flows with core Mach numbers ranging from 0.20 to 0.60. Because of the theory used in its derivation, the model is intended to work well outside of the range of the experiments. However, there is one caveat: past experience indicates that the model does not work when the vortex generators are placed in regions of sonic flow.

`ENDVORTEX`

Ends the vortex generator input block

Example

The following example illustrates the use of the `VORTEX GENERATOR` input block for one vortex generator array located between zones 2 and 3. The interface between zones 2 and 3 corresponds to the i_{max} boundary of zone 2 and the i_1 boundary of zone 3. The array contains two vortex generators, both mounted on the j_{max} viscous wall boundary of the zone interface with z coordinate values of -0.52 inches and 0.52 inches. The generators have the same geometric parameters: the chord length is 1.6 inches, the height is 0.4 inches, and the angle of incidence is 16 degrees.

```
VORTEX GENERATOR
  ZONE 2 BOUNDARY IMAX
  ZONE 3 BOUNDARY I1
  NUMBER 2
    JX ZLOC -0.52 1.6 0.4 16.0
    JX ZLOC  0.52 1.6 0.4 16.0
ENDVORTEX
```

See Also: BOUNDARY TVD

VISCOSITY — Specification of viscosity law

```
VISCOSITY {SUTHERLAND | WILKE | KEYE | CONSTANT vis}
```

This keyword allows you to specify the method of computing the transport properties. The equations shown below are for the laminar viscosity coefficient μ. For all the options except `WILKE`, in WIND the laminar thermal conductivity coefficient k is equal to the viscosity coefficient, when non-dimensionalized. For `WILKE`, the form of the equations used for k is the same as those used for μ, but with different constants from the chemistry data (*.chm*) file.

`SUTHERLAND` Use Sutherland's law, designed for ideal gases with $T > 180$ °R, as follows:

$$\mu = 2.329 \times 10^{-8} \frac{T^{3/2}}{T + 216}$$

`WILKE` Use Wilke's law, designed for multi-species flow (real gases). First, the viscosity coefficient is computed for each individual species n using Sutherland's law, as follows:

$$\frac{\mu_n}{\mu_0} = \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + S}{T + S}$$

where T is the local static temperature, and μ_0, T_0, and S are constants read from the chemistry data (*.chm*) file for species n. For N total species, the individual viscosity coefficients are combined using

$$\mu = \sum_{i=1}^{N} \frac{X_i \mu_i}{\sum_{j=1}^{N} X_j \phi_{i,j}}$$

where $\phi_{i,j}$ is a mixing coefficient computed as

$$\phi_{i,j} = \left[8\left(1 + \frac{M_i}{M_j}\right)\right]^{-1/2} \left[1 + \left(\frac{\mu_i}{\mu_j}\right)^{1/2} \left(\frac{M_j}{M_i}\right)^{1/4}\right]^2$$

X is the species mole fraction, and M is the species molecular weight.

`KEYE` Use Sutherland's law for $T \geq 180$ °R, Keyes' law for $T \leq 160$ °R, and a linear combination for 160 °R $< T <$ 180 °R. Sutherland's law is written as above. Keyes' law is given by:

$$\mu = 2.32 \times 10^{-8} \frac{T^{1/2}}{1 + (220/T) \times 10^{-9/T}}$$

And the linear combination is given by

$$\mu = f\mu_S + (1 - f)\mu_K$$

where μ_S and μ_K are the viscosity coefficients from Sutherland's and Keyes' laws, and $f = (T - 160)/20$.

`CONSTANT` *vis* Use a constant molecular viscosity of *vis* (slug/ft-sec)

In all of the above equations, μ is in slug/ft-sec and T is in °R.

WALL FUNCTION — Specify the use of wall functions

```
WALL FUNCTION {on|off} [zone_selector]
```

For turbulent flows, this keyword may be used to invoke wall function boundary conditions on viscous walls, using the White-Christoph law of the wall. Wall function boundary conditions allow calculations to be performed with fewer grid points, and generally with higher CFL numbers.

A wall function boundary must lie on one of the boundaries of the selected zone, and may be an overlapped or internal boundary. The first grid point adjacent to the boundary must be within the log layer (roughly $15 < y^+ < 100$). The recommended y^+ value at the first point off the wall is about 50. The boundary condition will revert to a no-slip condition when the first grid point off the wall falls below $y^+ = 15$, indicating that it is within the laminar sublayer.

Note that the wall function boundary condition is used to eliminate the grid points that would otherwise be required to resolve the laminar sublayer and the logarithmic layer. The grid beyond the first point off the wall should be the same as for a run without wall functions, in order to properly resolve the rest of the boundary layer. The *cfsubset* utility may be useful when modifying an existing grid for use with wall functions.

The implementation of wall functions in WIND involves only the modification of the wall flux, and does not reset the values of any flow quantities.

Viscous forces on wall function boundaries calculated using the LOADS keyword should only be used for determining convergence. Correct values should be calculated from the *.cfl* file using the integrate force command in CFPOST.

Based on limited test cases, the use of wall functions is not recommended if accurate predictions of wall heat transfer are required, or for flows with strong shock waves.

WALL SLIP — Iterations until no slip

`WALL SLIP [ITERATIONS]` *val* `[`*zone_selector*`]`

To minimize transients at the start of a WIND calculation, the velocity at no-slip boundaries is actually reduced from its initial value to the no-slip condition over a number of iterations. The `WALL SLIP` keyword allows the user to specify the number of iterations, given by *val*, on a zonal basis. The default value is 50 iterations.

WALL TEMPERATURE — Specify wall temperature

```
WALL TEMPERATURE [ADIABATIC | value [EXTRAPOLATE [SPECIES]]] [zone_selector]
```

This keyword allows you to specify a wall temperature on a zonal basis.

`ADIABATIC`	Use an adiabatic wall boundary condition for temperature. This is the default.
value	Use a constant wall temperature equal to *value* °R
`EXTRAPOLATE [SPECIES]`	Use extrapolation (zero gradient) for chemical species at the wall (finite rate chemistry only)

See Also: TTSPEC

11 Test Options

Several user-controlled options have been provided as an aid to modifying the WIND code. These options all may be selected by using the keyword `TEST` in the input data file. The test options typically control program features which are under test, and have not been accepted for production use. Each test option is of the form

`TEST` *number mode*

If *mode* is not described for a given test option, the user should use *mode* = 1 to activate that option.

As test options are accepted, they are "hard wired" into the code and the test option described here becomes meaningless. Test options may also be rejected based on trial runs. In that case, the option code described here will also become meaningless. For this reason, the list of valid options is not consecutive. Selection of an invalid option will be accepted by the WIND code, but will have no effect.

The various test options and modes are described in Table 3. For each option, the subroutines referencing that option are listed in parentheses.

Table 3: Non-Production Test Options

number	**Description**

1	Memory size check. Stop the run after displaying the memory requirements but before memory is allocated. (`mpinit, mpterm, opngrd, wind, zinit`)
	mode — Result
	$n < 0$ — Retrieve dimensions from the actual grid (*.cgd*) file
	$n > 0$ — Retrieve dimensions from a text file with the following format:
	`#zones`
	`idim1 jdim1 kdim1 nfrpts1`
	`...`
	`idimn jdimn kdimn nfrptsn`
	Each of the `nfrpts` values is 0 except for overlapping grids. This mode allows one to determine how much memory WIND will require before generating the grid. Note that the name of the text file should end in *.cgd*, like the actual grid file.

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
2	Designed for parallel processing data transfer debugging *mode* Result 1 Not used 2 Don't read zonal boundary data (`evrwbd`) 4 Don't solve zone (`evsolv`) 8 Don't write zonal data (`evwzon`) 16 Don't write zonal boundary data (`evrwbd`) 32 Not used 64 Not used 128 Not used 256 Don't update boundary conditions (`lpschm`) Set *mode* equal to the sum of the desired actions.
3	Parallel processing task tracing *mode* Action 1 Trace event reads/writes (`rwev`) 2 Trace file I/O (`rwnh, rwstat, rwd, rwi, rwr`) 4 Trace network traffic (`psexit, psrwev, psrwgv, psrwnh, psrwd, psrwi, psrwr`) 8 Trace task begin/end (`psspwn, tskbeg, tskidl`) 16 Print task queue for debugging (`psqprt`) Set *mode* equal to the sum of the desired actions. I.e., setting *mode* = 5 will trace both event reads/writes and network traffic.
4	Reserved for internal Boeing use in routines for hybrid unstructured grid capability (`opngrd`)
5	Compute the L2 norm of the residual for the Navier-Stokes equations in the same way as the "old" code (`l2norm`)
6	Write *.cfl* file compatible with the "old" code (`asnsx, asvisc, mpini2`)
7	Do not use high performance C I/O interface (`openf, rwstat`)
8	Use Version 2 common files (`mpinit, openf, rwstat`)
10	Print the time step information (i.e., minimum/maximum CFL and Δt) into the *.lis* file every *mode* cycles, instead of just on the first cycle. (`zsolv`)
17	Use "new" Baldwin-Barth turbulence model (`bbarth`)
20	Non-dimensionalize k and ω in the SST model the "old" way (`aijkrg, aikeps, sst, sstprtinp, sstpstprt`)

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
21	Spalart-Allmaras turbulence model. (See the TURBULENCE keyword.) (`redimsa, sabound, sapreinp, sinut, spalart`) *mode* — Result 0 — Use original 1992 model, with an f_{t2} term for laminar stabilization and a default freestream value for ν_t of 5.0. 1 — Like mode 0, except with "corrections" to the production and destruction terms. This is equivalent to the default model for WIND beta versions 4.15 to 4.92. 2 — Like mode 1, except without the f_{t2} term, and with a default freestream value for ν_t of 0.1. In addition, the initial value of the dependent variable $\tilde{\nu}$ is set to the freestream ν_t. This is equivalent to the default model for WIND versions prior to WIND beta 4.15, and includes a slight error that makes the model overly dissipative.
25	In the Baldwin-Lomax model, use y^+ based on wall vorticity (`blomax`)
26	Use local values in y^+ damping for the Baldwin-Lomax, Cebeci-Smith, Baldwin-Barth, and Chien k-ϵ models (`bbdamp, blomax, cebeci, kepy2`)
29	For the Cebeci-Smith model, use y^+ based on wall vorticity. (`algtur, cebeci`) For the Chien k-ϵ model, `TEST 29` is a production limiter (`kelhssch, keprod, kerhssch`) *mode* — Result 0 — Production limited to 20 × dissipation 1 — Production computed from vorticity, but not limited 2 — Production not limited 3 — Production computed from vorticity, then limited
30	Irrotational boundary condition at freestream inflow boundaries (`bcfree`)
40	Old switch for compressibility correction in k-ϵ model; use K-E COMPRESSIBILITY CORRECTION instead. (`keppstin`)
46	In the SST turbulence model, in blowing regions and bleed regions with a specified negative bleed flow rate, set $\mu_{turb} = 10\mu_{lam}$ along the wall. (`sstbound`)
47	For the algebraic turbulence models, smooth turbulent viscosity in each i-plane using simple averaging. *mode* = number of smoothing passes. (`smtvis`)
48	For the algebraic turbulence models, smooth turbulent viscosity in three dimensions using simple averaging. *mode* = number of smoothing passes. (`smtvis`)
49	Modified Runge-Kutta smoothing (see the SMOOTHING keyword) (`dampi, dampj, dampk, bdload`) *mode* — Result 2 — No pressure switch on second-order dissipation 3 — No pressure switch, and an LES type filtering of nonlinear terms

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
51	Limit the turbulent viscosity μ_T such that the maximum value of $\mu_T/(\mu_L)_\infty = mode \times 1000$. Suggested range is $50 < mode < 100$. Do not use this option with the Chien k-ϵ model; use `K-E MAXIMUM TURBULENT VISCOSITY` instead. (`keppstin, mutlim`)
52	When using `BLOW PLENUM`, print a warning when the plenum total pressure is automatically raised because it was less than the local static pressure (`bcbled`)
53	Allows no-slip in inviscid flow at "viscous wall" boundaries (`bcwall`)
54	Reserved for use at Boeing
55	Reserved for use at Boeing
56	No energy addition to fluid due to MFD equations (`emdef`)
57	Implicit terms on for the Spalart-Allmaras and SST turbulence models (`spalart, sst`)
58	Store the Lorentz force in the *.cfl* file instead of the electric field (`emdef`)
59	Apply `SMOOTHING` keyword values to the Spalart-Allmaras model as well as the mean flow solver. (Currently deactivated) (`spalart`)
61	When $mode = 2$, all boundary conditions are applied, whether or not they're consistent with the `IBLANK` values. This only affects corners, where there are usually multiple boundary conditions. So, if a wall boundary at $j = 1$ meets an outflow boundary at $i = i_{max}$, if `TEST 61 2` is specified, both boundary conditions are applied. (`tdbcgs`)
63	Eliminate the "fat" boundary cells in any coordinate direction. $mode = 1$, 2, or 4 indicates the i, j, and k direction, respectively. Set $mode$ equal to the sum of the desired directions. I.e., setting $mode = 5$ will eliminate the "fat" boundary cells in the i and k directions. (`tdarea`)
64	Remove `dt` from `dq` when computing residuals (`l2norm`)
65	In marching solutions, lower the CFL number for the last marching step (`zsolv`)
66	Don't update β in `gas3`, for `ireal` = 2 and `ispec` = 2. This test option is not recommended but will decrease run time. (`gas3`)
67	When solving turbulence model equations, treat bleed boundaries as slip walls instead of no-slip walls. This was the default behavior prior to WIND 5.101. (`kebc, sabound, sstbound`)
68	If the density is zero at a coupled boundary, issue a warning, ignore the coupling data, and continue. The default is issue an error message and abort. (`postrbs`)
70	Tolerance level for converging gas properties P, ρ, or T in the `gas` routines. Tolerance level = $(0.1)^{mode}$. (`gas1, gas2, gas3, gas4`)
71	Apply curve fit equations for thermodynamic properties even when the temperature is outside the applicable range. If $mode = 1$, an error message will be written whenever this occurs. For other non-zero values of $mode$, no error message is written. If this test option is not set, the run will be terminated if the temperature is outside the range of the curve fits. (`cpfun, gibfn, hfun, proper`)
75	?? (`prpold, tdimafk, tdimfu`)

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
77	Set bleed limits for a specific F/A-18 15% scale sting and distortion model, for use with the `BLEED AEDC` keyword (`bcaedc`) *mode* — Result 0 — F/A-18E 1 — F/A-18A
84	Use "old" viscous metric calculation (`dsolv, vismet`)
85	Check for zero volumes when computing viscous metrics (`dsolv, vismet`) *mode* — Result 1 — Check; if ≤ 0 print message and continue 2 — Check; if ≤ 0 print message and stop
87	Freezes supersonic inflow at initial conditions (`bcfree`)
88	Bypass negative T check in `tdgas1`. This is needed for chemistry if SHF (heat of formation) varies widely since we only have an old SHF to use to estimate T. (`lpschm, tdgas1`)
89	Use "old" species flux correction method (`gas1`)
90	Chemistry stuff (`chinv`) *mode* — Result 0 — Analytic chemistry Jacobian (`ns` $= 5$ only) 1 — Householder chemistry Jacobian (`ns` > 5) 2 — Solves chemistry source term explicitly
91	Gas constant (`chprtinp, therm1, thermp`) *mode* — Result 1 — $\beta = \gamma = \beta_\infty$ 2 — $\beta = \gamma = 1.4$
92	Utilize operator splitting for the reacting chemistry source terms to increase the stability of the integration, allowing more efficient solution of the coupled system. A 4th-order Pade approximation is used to integrate the reaction source terms, with *mode* setting the number of subiterations. Setting *mode* $= 0$ indicates no operator splitting. (`chimplicit, chrhss, prpold, rhssrc`)
93	Turn on/off chemistry species diffusion (`chrhsv, rhsvfl, tdutv1`) *mode* — Result 0 — Include all chemistry species diffusion terms 1 — Include chemistry species diffusion terms, except for the diffusion gradient in the conduction term 2 — Same as 0 3 — Ignore chemistry species diffusion
94	Turn off implicit chemistry terms (`tdimpl, tdutaa`)

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
95	Turn off chemistry source term (`rates, rates1, rates2, rates3, rates4, ratesa, ratesadl, ratesb, ratesf`)
96	Apply chemistry source term over *mode* iterations for finite-rate non-equilibrium chemistry (`chrhss`)
97	P. D. Thomas turbulence model scanning direction. By default, WIND starts at viscous walls and moves into the field. This test option forces the code to calculate turbulent parameters from *any* boundary, in addition to walls. (`algtur`) *mode* — Result 0 — use j lines 1 — use k lines 2 — use j and k lines
98	*mode* is the relaxation factor for the species mass fraction from the Liu and Vinokur (real gas) model. By increasing *mode*, the mass fraction relaxes faster. Default *mode* = 5. *mode* = 1 corresponds to instantaneous. (`gas1, gas2, gas3, gas4`)
99	Initialize finite rate chemistry with Liu and Vinokur curve fits (`gas2`) *mode* — Result 0 — Do not track the species (valid to 50K?) 1 — Track the species (valid to 10K?)
100	Characteristic time-stepping boundary condition (`bcfree`) *mode* — Result 0 — Second order, with limit of $\Delta\mathbf{Q} \leq \mathbf{Q}/2$ 1 — 1st-order, original characteristic treatment 2 — 2nd-order, original characteristic treatment 3 — 1st-order, Roe's average characteristic treatment 4 — 2nd-order, Roe's average characteristic treatment
102	Use time-averaged back pressure for mass flow boundary condition (`pdsmfr`)
104	Turn off implicit viscous terms (`tdutv1`)

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
105	Time step type (`prtinp, tdtmst`)

mode	Time Step Type
0	$\Delta t = \text{CFL}/\max(\lambda_\xi, \lambda_\eta, \lambda_\zeta)$
1	Flow angle scaling, $\Delta t = \text{CFL} \times (f_\xi \Delta\xi + f_\eta \Delta\eta + f_\zeta \Delta\zeta)$, where $f_\xi = \sqrt{1 + \tan\theta + \tan\psi}$ $f_\eta = f_\xi \tan\theta$ $f_\zeta = f_\xi \tan\psi$
2	Velocity scaling, $\Delta t = \text{CFL} \times \min(f_\xi \Delta\xi, f_\eta \Delta\eta, f_\zeta \Delta\zeta)$, where $f_\xi = u/u_\xi/\lvert u_\xi + c\rvert$ $f_\eta = u/u_\eta/\lvert u_\eta + c\rvert$ $f_\zeta = u/u_\zeta/\lvert u_\zeta + c\rvert$
3	$\Delta t = \text{CFL} \times \min(\Delta\xi, \Delta\eta, \Delta\zeta)/(\lvert u\rvert + c)$
4	$\Delta t = \text{CFL}/(\lambda_\xi + \lambda_\eta + \lambda_\zeta)$

106 Compute the time step at the start of every cycle (even when Newton time stepping is being used), instead of at the start of every iteration. (`axflow, lpgrp, lpmg, redim`)

108 Extrapolate freestream outflow (`bcfree`)

mode	Mach	Outflow Conditions
0	< 1	Hold upstream running characteristic at freestream
	> 1	Extrapolate all, even in boundary layer
1	All	Extrapolate all, even in boundary layer

109 Boundary flux treatment (`roewal, tdup1`)

For `tdup1`:

mode	Result
0	Conservative
1	Upwind extrapolation from interior

For `roewal`:

mode	Result
0	Characteristic inflow, conservation if flow parallel to wall
1	Characteristic regardless
1000	Use conservative wall treatment at all boundaries

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
110	Grid area variation limiting. Not allowed for `iorder` > 24, i.e., for the following Roe and Van Leer explicit operators: third-order fully upwind, fourth-order upwind-biased, fourth-order central, and fifth-order upwind-biased. (`roecof`) *mode* — A_2/A_1 0 — ∞ 1 — 2.0 2 — 1.5 3 — 1.33 4 — 1.1
111	Singular matrix check (`jacpr4, jacpr5, jacprg, tdsol4, tdsol9, tdsol11, tdsolg, tdsolv`) *mode* — Result 1 — Check, but don't print results 2 — Don't check
112	Corrected upwind scheme at boundaries. Defaults to corrected scheme, *mode* > 0 uses second order smoothing with *mode*/1000 as the smoothing level. Users should not use this option. (`rhsupw`)
113	Check for reverse flow at inflow and outflow boundaries (`bcconf, bcfree`) *mode* — Result 0, 1 — Print a warning message and continue 2 — Print an error message and stop
114	Central difference ζ operator (`tdup1`) *mode* — Result 0 — Upwind n — Central smoothing coefficient $= n/1000$
115	Do not rescale inviscid wall total velocity to equal adjacent value, just subtract the normal component from the adjacent value (`bcwall`)
116	Set inward pointing normal to zero in `tdbcm1` at unknown grid topology points (`tdbcm1`)
117	Freeze inflow boundaries, even in subsonic flow (`bcfree`) *mode* — Result 1 — Freeze all inflow 2 — Freeze only arbitrary inflow points 3 — Freeze characteristics on all $i = 1$ boundaries

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
118	Singular axis fix (`radavg`) *mode* — Result 0 — Average density, momentum components, and pressure 1 — Average density, velocity components, and pressure
121	Under-Relaxation of points adjacent to singular axis (`bcpinw, bcsing, kebc, relsng, sabound, sngthrm, sstbound`) *mode* — Result 0 — Value on axis is a radius-weighted average of the values at the adjacent points; values at the adjacent points are unchanged n — Value on axis computed as for mode 1; values at the adjacent points are computed from $F_{adj} = (1-r)F_{adj} + rF_{axis}$ where F_{adj} is the value at the adjacent point, F_{axis} is the axis value, and $r = n/1000$.
122	Allow left-handed coordinates (`tdarea`)
123	Track back pressure, mass flow, and integrated total pressure with outflow boundary conditions (`bcconf, pdsmfr`)
124	Write convergence information to list output (*.lis*) file every iteration instead of every cycle (`lpschm`)
126	Compressibility correction to Baldwin-Lomax turbulence model (`blomax`) *mode* — Result 0 — No compressibility correction 1 — $\kappa = 0.0180$ for Baldwin-Lomax model (CFL3D uses this)
127	Scale printed residual by maximum residual over all time steps (`lpschm`)
128	Check the L2 norm of the residual for convergence instead of the maximum residual (`l2norm`)
130	Roe scheme physical space extrapolation (`bcfree`)
131	For boundary layers on $j = 1$ walls, set the time step in the boundary layer to a (larger) "outer" value, defined as the value at $j = mode$. I.e., $(\Delta t)_j = (\Delta t)_{mode}$ for $j < mode$. (`tdtmst`)
132	Renormalize, changing from total to static values. Normalizing values in the *.cfl* file are unchanged. (`redim`)
133	Print multi-grid sub-iteration convergence data (currently deactivated) (`lpschm`)
134	Second order characteristic extrapolation for adjacent conditions (`bcfree`) *mode* — Result 2 — First order 0, 1 — Second order, using a minmod limiter

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
135	Resets the time step using a weighting function between the ordinary Euler CFL number and a new "viscous CFL number", for convergence acceleration in viscous layers. The viscous CFL number is set to *mode*/1000. Limited testing indicates that a value of *mode* = 50 is stable and increases the time step near the wall by at least an order of magnitude. (`tdtmst`)
136	Divergence checker, $mode = n_1 + 10n_2$, where (`lpgrp`) n_1 — Divergence Definition 1 — Max residual > 1.0, L2 norm increasing 2 — Max residual > 5.0, L2 norm increasing 3 — Max residual > 10.0, L2 norm increasing and n_2 — Action Taken When Diverging 1 — Terminate iteration for current cycle 2 — Abort run 3 — Reduce CFL number by 1/2 4 — Reduce CFL number by 1/2 and terminate iteration for current cycle
137	Butt line interpolation region for USERSPEC; smear `USERSPEC` conditions over 0.001 × butt line at minimum and maximum butt line (`uspeci`) *mode* — Result 0 — No interpolation n — $n = 0.001 \times$ butt line for interpolation
138	Use large cell Jacobians at boundaries (`bcwall, chrhsv, nsrhsv, tdarea, vismet`) *mode* — Result ≤ 1 — Use large cells 2 — Use large cells, central difference Jacobian 3 — Throw out half cell at boundaries 5 — Solve $\partial P/\partial n$ equation at walls
139	Turn on grid-based flux limiter (`tdup1`)
140	Use first-order differencing for computing $\partial(u, v, w)/\partial\xi$ term in vorticity used in turbulence models (`vortcy`)
141	2nd-order wall boundary conditions (explicit) (`bcwall`) *mode* — Result 1 — Second order $\partial p/\partial n$, $\partial T/\partial n$, and $\partial u_{tan}/\partial n$ 2 — Second order $\partial p/\partial n$ and $\partial T/\partial n$, but not $\partial u_{tan}/\partial n$
147	?? (currently deactivated) (`turpdt`)
148	?? (currently deactivated) (`turpdt`)

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
150	Singular axis on symmetry planes. When symmetry plane test fails, zero this component of velocity. (`bcsing`) *mode* — Result 1 — $u = 0$ 2 — $v = 0$ 3 — $w = 0$ 4 — do not zero any component (use average) 5 — $v = w = 0$ 6 — $u = w = 0$ 7 — $u = v = 0$
151	For a singular axis, the value on the axis is a radius-weighted average of the values two points away from the axis, instead of the values at the adjacent points; values at the adjacent points are set to the average of the axis value and the value two points away from the axis. E.g., for a singular axis at $j = 1$, with the k direction "circumferential", the value on the axis is a radius-weighted average of the values at $j = 3$, instead of at $j = 2$. Then for each k, the value at $j = 2$ is set to the average of the values at $j = 1$ and $j = 3$. This test option overrides `TEST 121`. (`bcpinw, bcsing, kebc, kerot, radavg, relsng, sabound, sngthrm, sstbound`)
153	Iteration frequency for updating pressure at outflow boundaries. Default is 5. (`bcconf`)
157	`USERSPEC` inflow (`uspeci`) *mode* — Result 1 — Put `USERSPEC` inflow at all points in the buttline range. Do not check for above/below vehicle. 2 — Same as mode 1, but also ignore fuselage station check.
158	Reserved for internal Boeing use in routines for hybrid unstructured grid capability (`opngrd, pstinp`)
160	Pressure correction factor = *mode*/1000 for specified mass flow boundary (`pdsmfr`)
162	Cebeci-Smith boundary layer edge definition (`cebeci`) *mode* — Result 0 — 1.0% change in U_{total} 1 — 0.995 H_t 2 — 0.99 U_{total} 3 — 0.9999 U_{total}

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
163	Criteria for defining F_{max} in Baldwin-Lomax model *mode* / Result > 0 Search outward from wall; use first peak in F that is followed by a fractional decrease in F of $mode/1000$. < 0 Use the max F value between the wall and the $\|mode\|$'th grid point from the wall The default is +700. (`blomax`)
164	Iteration interval for updating gas properties and species for `ireal` = 1 and 2. The default value is 1. (`tdgas1`)
165	Sets the tolerance for the distance between grid points used to determine a singular direction to $10^{mode/1000}$. The default is a tolerance of 10^{-8} (i.e., $mode = -8000$). (`bcsing, direct`)
168	For the algebraic turbulence models, begin turbulent flow at $i = mode$ (`tdvis1`)
170	NASA Ames time step formula. (CFL increases as $1/\sqrt{\Delta y}$ near the wall. Thus, Δt decreases as $\sqrt{\Delta y}$, not Δy as the default.) C_A is scalar coefficient on CFL; i.e., $\text{CFL}_{wall} \propto C_A$. This test option has no effect if `TEST 105` mode 1, 2, or 3 is used. (`tdtmst`) *mode* / C_A 1 0.01 n $0.001n$
172	Turn off base energy for `ireal` = 1 (`aichem, aijkrg, gas1, gas2, gas3, gas4, gasint, pstinp, stomp, tdimfp, tdroe4, uspeci`)
174	For the algebraic turbulence models, the iteration interval for updating the turbulent viscosity. The default is 1. (`tdvis1`)
175	Boundary conditions at freestream radial outer boundaries (`nzn` = −6). (`bcfree`) *mode* / Result 0 Compute characteristics from freestream conditions 1 Compute characteristics from conditions at $i = 1$ along boundary 2 Extrapolate without testing at k boundaries; treat i and j boundaries as in mode 1
177	Freeze maximum residual (`lpgrp`)
178	Insert freestream species buffer during `BLOW VALVE` relaxation (`bcbled`)
179	Modify solidbody rotation radius to get linear swirl profile in r, but with zero velocity not at center of rotation. $mode = 1000r_0$, where r_0 is the radius (from the point x_c, y_c, z_c specified using the `SOLIDBODY` keyword in the `ARBITRARY INFLOW` keyword block) for zero velocity. (`rotat`)
180	Defines the radius of the solidbody rotation core for actuator disk free vortex modeling. $mode = 1000r_{core}$, where r_{core} is the solidbody core radius. (`rotat1`)
181	?? (`evrwbd, wbnd2`)

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
182	?? (`tdbcgs`)
185	?? (`gasint, mpinit`)
187	*mode*/1000 = factor for suppression of streamwise pressure gradient when marching. The default is 950, and values below 800 are not recommended. When separation or strong adverse pressure gradients are causing problems, values between 800 and 900 will really help. (`rhsmar, tdimfp`)
189	If a first-order upwind explicit operator modified for stretched grids is used (e.g., `RHS ROE FIRST PHYSICAL`, then `TEST 189 1` must also be specified. (`prtinp`)
190	Outgoing wave Roe boundary treatment (`pstinp, roecof, roeht, tdbcgs`) *mode* — Result 0, 1 — Use normal Roe boundary treatment (uses boundary point in formulation) 2 — Lower order by one (does not use boundary point in formulation). This option cannot be used with `TVD` in the same zone. 3 — Use zero-order extrapolation
191	Options used with the `BLOW` and `BLOW SURFACE` keywords. *mode* — Result 0 — Angles are defined with respect to the surface normal. 1 — Angles are defined with respect to the projection of the surface normal onto a constant z plane. 2 — Angles are defined as in mode 0, and equilibrium air chemistry is allowed. The surface values of the effective gamma and compressibility factor are approximated by using the values at the adjacent flowfield point. 3 — Angles are defined as in mode 1, and equilibrium air chemistry is allowed. 4 — Backward compatibility mode for the `BLOW` keyword. With this mode, the syntax is "`BLOW` *ibreg mdot* T_{inj} *angle*", where *angle* is the blowing angle relative to the x-y plane, in degrees. Note, though, that this mode is not recommended, as it does not provide the requested mass flux for surfaces whose normal vector contains components in the z direction (i.e., surfaces with transverse curvature). Modes 0 and 1 apply to both the `BLOW` and `BLOW SURFACE` forms; modes 2 and 3 only apply to the `BLOW SURFACE` form; and mode 4 only applies to the basic `BLOW` form. (`bcbled`)
192	Save metrics in a temporary file. After the first cycle, metrics will be read rather than computed. This eliminates the CPU resources required to re-compute the metrics each cycle, but adds significant I/O to the computation. In the past, on at least some Cray systems, this reduced the CPU time by approximately 40.8 μsec/(node-cycle). On the more common platforms, however, it is generally faster to re-compute the metrics rather than store them. (`lpcycl`)

Continued on next page

Table 3: Non-Production Test Options (*Continued*)

number	Description
193	Do *not* stop if a singular line is encountered normal to a wall (`bbdamp, blinit, kepy2`)
194	Bypass singular viscous metric check (`dsolv, vismet`)
195	When using `BLOW SURFACE`, print a warning when the flowfield static pressure becomes larger than the plenum total pressure, causing blowing to be turned off at that point. Note that this is a five-line message written for each iteration and each "closed" node, and could cause the *.lis* file to become very large very quickly. (`bcbled`)
196	Overlapping grid: print an error message if there are no points adjacent to a fringe point (`inttur, tdbcgs`)
197	Roe self-coupling mode (`pstinp`) *mode* — Result 0 — Once per iteration, using `bcself` 1 — Once per cycle, using standard zone coupling
198	Sets the tolerance level for fuselage station `USERSPEC` inflow to *mode*/1000. The default tolerance is 0.01 (i.e., *mode* = 10). (Currently deactivated) (`uspeci`)
199	Singular axis averaging — average from 1 to ($max - 1$), *not* 1 to *max*. (`bcsing, linzero, radavg, relsng, sngthrm`)
200	Don't bomb for negative speed of sound in `tdroe3` (`tdroe3`)

References

Baldwin, B. S., and Lomax, H. (1978) "Thin Layer Approximations and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257.

Baldwin, B. S., and Barth, T. J. (1990) "A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows," NASA TM 192847.

Bush, R. H. (1988) "A Three Dimensional Zonal Navier-Stokes Code for Subsonic Through Hypersonic Propulsion Flowfields," AIAA Paper 88-2830.

Chien, K. Y. (1982) "Prediction of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model," AIAA Journal, Vol. 20, No. 1, pp. 33–38.

Chung, J., and Cole, G. L. (1996) "Comparison of Compressor Face Boundary Conditions for Unsteady CFD Simulations of Supersonic Inlets," NASA TM 107194.

Cornell, W. G. (1958) "Losses in Flow Normal to Plane Screens," Transactions of the ASME, May 1958, pp. 791–799.

Mani, M., Ladd, J. A., Cain, A. B., and Bush, R. H.(1997) "An Assessment of One- and Two-Equation Turbulence Models for Internal and External Flows," AIAA Paper 97-2010.

Mayer, D. W., and Paynter, G. C. (1994) "Boundary Conditions for Unsteady Supersonic Inlet Analyses," AIAA Journal, Vol. 32, No. 6, pp. 1200–1206.

McLafferty, G., and Ranard, E. (1958) "Pressure Losses and Flow Coefficients of Slanted Perforations Discharging from Within a Simulated Supersonic Inlet," United Aircraft Corporation, Report R-0920-1, Dec. 1958.

Menter, F. R. (1993) "Zonal Two Equation k-ω Turbulence Models for Aerodynamic Flows," AIAA Paper 93-2906.

Nichols, R. H., and Tramel, R. W. (1997) "Application of a Highly Efficient Numerical Method for Overset-Mesh Moving Body Problems," AIAA Paper 97-2255.

Paynter, G. C. (1998) "Modeling the Response from a Cascade to an Upstream Convective Velocity Disturbance," AIAA Paper 98-3570.

Rodi, W., and Scheuerer, G. (1986) "Scrutinizing the k-ϵ Turbulence Model Under Adverse Pressure Gradient Conditions," Transactions of the ASME Journal of Fluids Engineering, Vol. 108, pp. 174–179.

Romer, W. W., and Bush, R. H. (1993) "Boundary Condition Procedures for CFD Analyses of Propulsion Systems — The Multi-Zone Problem," AIAA Paper 93-1971.

Sajben, M. (1999) "Prediction of Acoustic, Vorticity and Entropy Waves Generated by Short-Duration Acoustic Pulses Incident on a Blade Row," ASME Paper 1999-GT-148.

Shur, M., Spalart, P. R., Strelets, M., and Travin, A. (1999) "Detached-Eddy Simulation of an Airfoil at High Angle of Attack," Fourth International Symposium on Engineering Turbulence Modeling and Measurements, May 24–26, 1999, Corsica.

Slater, J. W., and Paynter, G. C. (2000) "Implementation of a Compressor Face Face Boundary Condition Based on Small Disturbances," ASME Paper 2000-GT-0005.

Spalart, P. R., and Allmaras, S. R. (1992) "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439.

Spalart, P. R., Jou, W. H., Strelets, M., and Allmaras, S. R. (1997) "Comments on the Feasibility of LES for Wings, And on a Hybrid RANS/LES Approach," First AFOSR International Conference On DNS/LES, Aug. 4–8, 1997, Ruston, Louisiana. In *Advances in DNS/LES*, Liu, C., and Liu, Z., eds., Greyden Press, Columbus, Ohio.

Syberg, J., and Hickox, T. E. (1972) "Design of a Bleed System for a Mach 3.5 Inlet," NASA CR-2187, Sept. 1972.

Thomas, P. D. (1979) "Numerical Method for Predicting Flow Characteristics and Performance of Nonaxisymmetric Nozzles — Theory," NASA CR 3147.

Tramel, R. W., and Nichols, R. H. (1997) "A Highly-Efficient Numerical Method for Overset-Mesh Moving-Body Problems," AIAA Paper 97-2040.