

RVC3D - Rotor Viscous Code 3-D

User's Manual and Documentation

Version 300, November 15, 2000

Dr. Rodrick V. Chima
NASA Glenn Research Center, MS 5-10
21000 Brookpark Road
Cleveland, Ohio 44135 USA

phone: 216-433-5919
fax: 216-433-5802
email: fsrod@grc.nasa.gov
internet: <http://www.grc.nasa.gov/WWW/5810/webpage/rvc.htm>

Abstract

RVC3D (Rotor Viscous Code 3-D) is a computer code for analysis of three-dimensional viscous flows in turbomachinery. The code solves the thin-layer Navier-Stokes equations with an explicit finite-difference technique. It is applicable to stationary linear cascades or annular blade rows with or without rotation about the x-axis. Two algebraic turbulence models and a simple tip clearance model are included. The code has been tested on numerous fan and turbine blades and has been used heavily at NASA Lewis Research Center for fan analysis and design and analysis of turbine endwall heat transfer. The code may be run on a Cray computer with run times of one to several hours depending on grid size and flow characteristics, or on workstations with longer run times. This report serves as the user's manual and documentation for the RVC3D code. The code and some aspects of the numerical method are described. Steps for code installation and execution are given for both Cray computers and workstations. The grid, input, and output variables are described in detail.

Introduction

A three-dimensional Navier-Stokes code has been developed for analysis of isolated turbomachinery blade rows. The code, called RVC3D (Rotor Viscous Code 3-D,) was originally described in (1) where results for a blunt fin problem and an annular turbine cascade were given, in (2) where results for a transonic fan were given, and in (3) where endwall heat transfer results were given. These references describe the mathematical formulation of the RVC3D code and should be cited in any publication resulting from the use of the code. A brief description of the code is given below.

The code solves the Navier-Stokes equations formulated in a Cartesian coordinate system with rotation about the x-axis. The equations are mapped to a general body-fitted coordinate system. Streamwise viscous terms are neglected using the thin-layer assumption, but all cross-channel viscous terms are retained. Turbulence effects are modeled using either a 3-D adaptation of the Baldwin-Lomax turbulence model (4) or an adaptation of the Cebeci-Smith model (3). The equations are discretized using second-order finite-differences and solved using a multistage Runge-Kutta scheme. The user may choose the number of stages in the scheme, but a four stage scheme is recommended. A spatially-variable time-step and implicit residual smoothing are used to accelerate convergence. Preconditioning may also be used to accelerate convergence for low-speed (incompressible) flows.

C-type grids are used to give good resolution of blade leading-edges and wakes (see figure 1.) Grid input is in standard PLOT3D xyz-file format, so any C-grid generator can be used. However two grid codes have been developed specifically for use with RVC3D, STACK and TCGRID. STACK reads a 2-D grid generated by the GRAPE code (7), and generates a 3-D grid for a linear or annular blade row by stacking the 2-D grid spanwise. TCGRID (6) is a general 3-D C- or H-grid generator for

k	α^1	α^2	α^3	α^4	α^5	λ^*
2	1.2	1.				.95
3	.6	.6	1.			1.5
4	.25	.3333	.5	1.		2.8
5	.25	.1667	.375	.5	1.	3.6

Table 1 - Runge-Kutta parameters $\alpha^1 - \alpha^5$ and maximum Courant number λ^* for k stage schemes.

turbomachinery. It reads annulus and blade geometry in either MERIDL format or NASA Lewis compressor design code format. It generates C-type grids at several spanwise locations using a version of the GRAPE code, then reclusters the grids spanwise.

RVC3D is written completely in Fortran and runs as a batch job on most workstations or mainframe computers. It has even been run on a PC. Solution times range from one to several hours on a Cray C-90 or usually overnight on workstations, depending on grid size and flow characteristics.

Namelist input data must be supplied to RVC3D as an ascii dataset. Printed output consists of a residual history, spanwise profiles of blade-to-blade averaged quantities at the grid inlet and exit, and streamwise profiles of various quantities on the blade surfaces. No graphical output is provided, but the solution files can be read directly and plotted using the public domain CFD visualization codes PLOT3D and FAST, or the commercial codes FIELDVIEW and TECPOT.

This documentation briefly describes how the RVC3D code works. Instructions for dimensioning, compiling, and running RVC3D are given for Silicon Graphics (SGI) workstations and Cray mainframes. The namelist input variables are described in detail. Finally, the structure of the output file is described.

Numerical Method

Multistage Runge-Kutta Scheme

Multistage schemes were developed by Jameson, Schmidt, and Turkel (8) as a simplification of classical Runge-Kutta integration schemes for ODE's. The simplification reduces the required storage, but also reduces the time-accuracy of the schemes, usually to second order. The following discussion of these schemes should give some guidance in choosing parameters for running the code.

A k -stage scheme may be written as:

$$q^k = q^0 - \alpha^k \Delta t (R_I^k + R_V^0)$$

where q is an array of five conservation variables (see "Solution Q-File", pp. 13.), k is the stage count, q^0 is the previous time step, α^k are the multistage coefficients discussed below, Δt is the time step, R_I^k is the inviscid part of the residual, and R_V^0 is the viscous part of the residual including the artificial dissipation. Note that R_I^k is evaluated every stage, but R_V^0 is evaluated only at the initial stage for computational efficiency.

The maximum stable Courant number λ^* for an n -stage scheme can be shown to be $\lambda^* = n - 1$. The actual stability limit depends on the choice of α^k . For consistency α^n must equal 1, and for second-order time accuracy α^{n-1} must equal $1/2$. The values of α^k used in the code and the theoretical maximum Courant number λ^* are set by a **data** statement in subroutine setup and are given in table 1.

The number of stages is set with the variable *nstg*. Using *nstg* = 4 is recommended, although Jameson et. al. tend to favor 5 stages. The 5-stage scheme in RVC3D is set up to calculate the artificial and physical dissipation every other stage. This tends to be more robust than the standard 4-stage scheme, but also very expensive. The 5-stage scheme should only be used if the 4-stage scheme will not run.

A spatially-varying Δt is used to accelerate the convergence of the code. Setting *ivdt* = 1 causes the Courant number to be set to a constant (input variable *cfl*) everywhere on the grid, and to be recalculated every *icrnt* iterations. The spatially-varying Δt option is highly recommended. Set *icrnt* to a moderate number, e.g. 10, so that the time step is recalculated occasionally. The time step is recalculated when the code is restarted and may cause jumps in the residual if *icrnt* is too big.

Implicit residual smoothing (described later) may be used to increase the maximum Courant number by a factor of two to three, thereby increasing the convergence rate as well.

Artificial Viscosity

The code uses second-order central differences throughout and requires an artificial viscosity term to prevent odd-even decoupling. A fourth-difference artificial viscosity term is used for this purpose. This term is third-order accurate in space and thus does not affect the formal second-order accuracy of the scheme. The input variable *avisc4* scales the fourth-difference artificial viscosity, and should be set between 0.25 and 2. Start around 1. If the solution is wiggly, increase *avisc4* by 0.5. If it is smooth, try reducing *avisc4* by 0.5. Larger values of *avisc4* may improve convergence somewhat, but the magnitude of *avisc4* has little effect on predicted losses or efficiency.

The code also uses a second-difference artificial viscosity term for shock capturing. The term is multiplied by a second difference of the pressure that is designed to detect shocks. Note that the second-difference artificial viscosity is first-order in space, so that the solution reduces to first-order accurate near shocks. Two other switches developed by Jameson (8) are used to reduce overshoots around shocks. The input variable *avisc2* scales the second difference artificial viscosity, and should be set between 0. and 2. Use 0. for purely subsonic flows, and start with 1. for flows with shocks. If shocks are wiggly, increase *avisc2* by 0.5. If they are smeared out, try decreasing *avisc2* by 0.5. Shocks will be smeared over four or five cells regardless of the value of *avisc2*. The magnitude of *avisc2* also has little effect on predicted loss or efficiency.

Eigenvalue scaling described in (2) is used to scale the artificial viscosity terms in each grid direction. This greatly improves the robustness of the code. The artificial viscosity is also reduced linearly by grid index near walls to reduce its effect on the physical viscous terms. Input variables *jedge*, *kedgh*, and *kedgt* are the indices where the linear reduction begins.

A first-order artificial viscosity term may be added to smooth the solution drastically during solution start-up. The variable *avisc1* scales this term. First-order artificial viscosity will greatly improve the convergence rate while greatly diminishing the accuracy of the solution. It will thicken boundary layers, smear shocks, and greatly increase predicted loss. Do not use first-order artificial viscosity except to start a new solution. A warning is printed in the output when *avisc1* > 0.

Implicit Residual Smoothing

Implicit residual smoothing was introduced by Lerat in France and popularized by Jameson in the U.S. as a means of increasing the stability limit and convergence rate of explicit schemes. The idea is simple: run the multistage scheme at a high, unstable Courant number, but maintain stability by smoothing the residual occasionally using an implicit filter. The scheme can be written as follows:

$$(1 - \varepsilon_{\xi} \delta_{\xi\xi})(1 - \varepsilon_{\eta} \delta_{\eta\eta})(1 - \varepsilon_{\zeta} \delta_{\zeta\zeta})\bar{R} = R$$

where ε_{ξ} , ε_{η} , and ε_{ζ} are constant smoothing coefficients in the three body-fitted coordinate directions ξ , η , and ζ indicated in figure 2. Here δ is a second-difference operator, \bar{R} is the smoothed residual, and R is the unsmoothed residual.

It can be shown that if the scheme converges implicit residual smoothing does not change the solution. Linear stability theory shows that the scheme can be made unconditionally stable if ε_i is big enough, but also shows that the effects of the artificial viscosity are diminished as the Courant number is increased. In practice the best strategy seems to be to double or triple the Courant number of the unsmoothed scheme. If the residual is smoothed after every stage, the theoretical 1-D values of ε_i needed for stability are given by:

$$\varepsilon_i \geq \frac{1}{4} \left[\left(\frac{\lambda}{\lambda^*} \right)^2 - 1 \right]$$

where $\hat{\lambda}^*$ is the Courant limit of the unsmoothed scheme (given in the previous table,) and λ is the larger operating Courant number. For example, to run a four-stage scheme at a Courant number $\lambda = 5.6$, the smoothing coefficient should be:

$$\varepsilon_i \geq \frac{1}{4} \left[\left(\frac{5.6}{2.8} \right)^2 - 1 \right] = 0.75$$

A single variable $eps = \varepsilon$ is input to RVC3D. The 1-D limit given above usually gives a reasonable estimate for ε , but the code will converge best when ε is minimized. Values of ε_ε , ε_η , and ε_ζ are evaluated at each grid point within the code by scaling eps using the same Eigenvalue scaling coefficients used for the artificial dissipation. This has proven to be quite robust.

In rare cases it may be necessary to increase the residual smoothing coefficient in a particular direction. This can be accomplished using input variables epi , epj , and epk , which are constants (usually 1.) that multiply ε_i at each point.

Implicit residual smoothing involves a scalar tridiagonal inversion for each variable along each grid line in each direction. It adds about 20 percent to the cpu time when applied after each stage. Smoothing can be done after every other stage to reduce cpu time (about 7 percent,) but eps must be increased (approximately doubled.)

Recommended starting values are: $nstg = 4$, $cfl = 5.6$, $irs = 1$, and $eps = 0.75$. If the code blows up quickly try increasing eps to 1.5. Very large values of eps (e.g. > 3) may stabilize a stubborn calculation but prevent the residuals from decreasing. If the residuals drop a little then climb to a large, constant value, eps is probably too big and the solution is probably incorrect.

Preconditioning

Density-based schemes like RVC3D solve the continuity equation by driving the density residual to zero. For low speed (nearly incompressible) flows the density residual is naturally near zero, and the schemes fail to converge. Preconditioning, described by Turkel in ref. (5) improves the convergence rate in two ways. First, it replaces the q-variables $q = [\rho, \rho u, \rho v, \rho w, e]$ with variables that are better-behaved at low speeds $W = [p, \rho u, \rho v, \rho w, h]$, where p is the pressure and h is the total enthalpy. Second it multiplies the equations by a matrix designed to equalize the wave speeds of each equation. The preconditioning matrix has the local flow velocity in the denominator and must be limited when the velocity becomes small. The preconditioning operator is designed so that it has no effect on the steady-state solution.

Preconditioning works extremely well for the Euler equations and less well for the Navier-Stokes equations. It will allow solutions at very low speeds that simply would not work otherwise.

Compiling and Running RVC3D

RVC3D is supplied as a unix script which generates the source and include files and compiles them. The format of the script file is shown below.

```
#!/bin/csh -f
cat > rvc3d.f << `'/eof'
#RVC3D source code goes here
`'/eof'
#-----
cat > noncray << `'/eof'
#two replacements for Cray-specific routines go here
`'/eof'
#-----
cat > csca << /eof
#scalar common block goes here
eof
#-----
#other common blocks follow
#-----
cat > param << /eof
#code dimensioning parameters go here
parameter(ni=165,nj=34,nk=33)
/eof
#-----

#compiler commands with options go here
/bin/rm...
```

On a unix platform, edit the script and go to the bottom. Change the parameter statements to values greater than or equal to the size of the grid to be run. (See *Parameter Statements* below.) Comment, uncomment, or add compilation commands appropriate for the computer to be used. (See below for compilation commands for SGI and Cray computers.) Save the script, set execute permission, and execute it.

On a PC, manually strip out, save, and compile the files between the *cat* and */eof* commands.

Parameter Statements

RVC3D uses a parameter statement to make redimensioning simple. The parameter statement and labeled common blocks are inserted during compilation using Fortran include statements. A typical parameter statement is shown above. Actual parameter values may be different in the distribution code. RVC3D checks the user input against the dimensioning parameters and stops with a fatal error message if the code is not dimensioned properly.

For a C-grid dimensioned (im, jm, km), the dimension parameters must have $ni \geq im$, $nj \geq jm + 1$, and $nk \geq km$. The j-direction requires an extra line of storage for a dummy grid line used in RVC3D to implement the periodic boundary conditions.

Thirty variables are stored at each grid point. Thus the computer memory required to run RVCQ3D is $30 \times ni \times nj \times nk$ words for arrays, plus about 200K words for the executable code.

Compiling RVC3D on Silicon Graphics Workstations (IRIX)

The following commands can be used to compile RVC3D on various SGI workstations:

```
#SGI power series compiler
f77 -pfa -O2 -lfpe -o rvc3d rvc3d.f
strip rvc3d

#SGI R8000 CPU
f77 -O3 -mips4 -WK, -o=0, -so=2, -ro=0 -OPT:round=3:IEEE_arith=3 \
-lfastm -o rvc3d rvc3d.f noncray.f
```

```
strip rvc3d

#SGI R4000 CPU (Indigo 2)
f77 -O2 -sopt -mips2 -lfpe -o rvc3d rvc3d.f
strip rvc3d
```

Compiling RVC3D on Cray Research Computers (UNICOS)

The following command will compile RVC3D on a Cray Y-MP or C-90:

```
cf77 -Zc -Wf" -o aggress -M367 -es" -o rvc3d rvc3d.f
```

The `-es` option produces a compiler listing, and the `-M367` option omits the *include* statements from the listing. RVC3D will autotask (run on multiple processors) on a Cray when compiled as follows:

```
cf77 -Zu -o rvc3d rvc3d.f
```

Running RVC3D

The executable program is run as a standard unix process:

```
rvc3d < std_input > std_output &
```

Standard Input, Output, and Binary Files

An ascii input file for RVC3D is read from Fortran unit 5 (standard input.) Printed output from RVC3D is written to Fortran unit 6 (standard output.) Binary files linked to Fortran units 1-3 may be used in the execution of RVC3D, depending on input options. The units are used as follows:

- fort.1 input grid file (see "Grid XYZ-File" pp. 13), required
- fort.2 input solution file (see "Solution Q-File" pp. 13), read if *iresti* = 1
- fort.3 output solution file (see "Solution Q-File" pp. 13), written if *resto* = 1

All files are treated as unformatted binary files. They are not explicitly *opened* in the code. Files are usually linked to file names before running RVC3D,

```
ln input.grid.file fort.1
ln input.solution.file fort.2
ln output.solution.file fort.3
```

Binary grid files can be used immediately on the same type of computer on which they were generated. Files generated on an SGI machine can be read into PLOT3D using the *read/unformatted* option. Files generated on a Cray can be converted to SGI format in one of two ways:

1. By using the *itrans* command at NASA Ames or the *irisbin* command at NASA Lewis to convert the files to SGI binary. Files converted using *itrans* or *irisbin* can be read into PLOT3D using the *read/binary* option <default>.

```
itrans file.xyz file.SGI.xyz
irisbin -u -v file.xyz file.SGI.xyz
```

2. By *assigning* the files as 32 bit ieee binary files on the Cray before execution. The lower precision does not affect the accuracy of RVC3D. Files written on a Cray while assigned as ieee binary can be used directly on an SGI machine and can be read into PLOT3D using the *read/unformatted* option.

```
assign -F f77 -N ieee fort.1
assign -F f77 -N ieee fort.2
#etc.
```

RVC3D Input

Defaults are given in angle brackets, <Default=value> or <default.> If no default is given the value MUST be input.

Title

ititle An alphanumeric string of 80 characters or less printed to the output. The character string must be enclosed in single quotes. The following FORTRAN input statement is used to read *ititle*:

```
read *,ititle
```

&nam1 - Grid Size Parameters

im Grid size in i- (streamwise) direction. Must agree with *im* read from the grid file. The code must be dimensioned with $ni > im$.

jm Grid size in j- (blade-to-blade) direction. Must agree with *jm* read from the grid file. The code must be dimensioned with $nj > jm + 1$ to allow for a dummy grid line added internally to implement the periodic boundary conditions.

km Grid size in k- (spanwise) direction. Must agree with *km* read from the grid file. The code must be dimensioned with $nk > km$.

itl i-index of lower trailing-edge point on a C-grid. *itl* is printed in the TCGRID output. *itl* can be 1 for a blade that extends to the exit of the grid.

iil i-index of the last periodic point on the outer boundary. Also acts as the lower point on the inlet boundary. *iil* is printed in the TCGRID code output.

ktip k-index of the blade tip. Tip clearances are modeled in RVC3D by forcing the flow to be periodic across the blade thickness for $k > ktip$. *ktip* should be chosen as the grid line closest to the actual blade tip. *ktip* can be determined from the TCGRID output or from the RVC3D spanwise output. < 0 disables the clearance model>

&nam2 - Algorithm Parameters

nstg Number of stages for the Runge-Kutta scheme, usually 4, but can be 2-5. <4>

cfl Courant number, typically 5.6 (see “Multistage Runge-Kutta Scheme” pp. 2.) If *ivtstp* = 0, *cfl* is the maximum Courant number, usually located somewhere near the leading edge at the blade surface. If *ivtstp* = 1, the Courant number will equal *cfl* everywhere. <5.>

avisc1 First-order artificial dissipation coefficient. Usually 0., but can be used to stabilize a solution that blows up at startup. Set *avisc1* = 1. for the first 50 or so iterations if necessary, but be sure to set *avisc1* = 0. as soon as the solution is running stably. (see “Artificial Viscosity”, pp. 3..) <0.>

avisc2 Second-order artificial dissipation coefficient. Typically 0. - 2. Use 0. for purely subsonic flow or 1. for flows with shocks. <0.5.>

avisc4 Fourth-order artificial dissipation coefficient. Typically 0.25 - 1.5. Start at 1.0 and reduce *avisc4* to 0.5 if possible. <0.5.>

irs Implicit residual smoothing flag. Usually = 1. (see “Implicit Residual Smoothing”, pp. 3.)

= 0 No residual smoothing

= 1 Implicit smoothing after every Runge-Kutta stage <default.>

= 2 Implicit smoothing after every other stage. *eps* must be increased for this option to work.

eps Overall implicit smoothing coefficient based on the 1-D stability limit (see “Implicit Residual Smoothing”, pp. 3.) RVC3D will calculate the 1-D limit if *eps* is defaulted.

epi, epj, epk	Implicit smoothing coefficient multipliers for the i , j , and k directions. (see “Implicit Residual Smoothing”, pp. 3.) Rarely used. <1.>
itmax	Number of iterations, typically 50-1000 per run, but 1000-3000 may be needed for a converged solution.
ivdt	Variable time step flag. = 0 Spatially constant time step. = 1 Spatially variable time step. <default, highly recommended>
ipc	Preconditioning flag, (see “Preconditioning” pp. 4.) = 0 No preconditioning. <default> = 1 Preconditioning using the Merkel, Choi, Turkel scheme. Should give a substantial speedup for Mach numbers < 0.3. = 2 Solves the equations using the preconditioning variable set, but sets the preconditioning matrix to the identity matrix. Used to debug the preconditioning routines.
pck	Constant limiter (Turkel’s parameter k) for preconditioning. The denominator in the preconditioning matrix is limited to be $> pck \times q_{ref}^2$. Typically 0.1 - 0.3, but larger values may be necessary for stability. <0.15>
refm	Reference relative Mach number used to find q'_{ref} described above. Should be approximately the largest Mach number expected in the flow. If the code blows up, try increasing $refm$ by 0.1. Convergence is mildly sensitive to $refm$ and pck , so try to keep these values as small as possible. <emxx>

&nam3 - Boundary Condition & Code Control

ibcin	Inlet boundary condition flag. In all cases P_0 , T_0 and v or v_θ are fixed at the inlet. For subsonic flow a Riemann invariant is extrapolated from the interior and $ibcin$ determines how u and w (for linear geometries) or v_r (for annular geometries) are determined. = 1 The inflow is aligned with the meridional grid direction. <default> = 2 Supersonic inflow - all quantities are fixed at the inlet. = 3 The ratio of w/u (or v_r/u) is specified. = 4 w or v_r is specified directly.
ibcex	Exit boundary condition flag. Four primitive variables are extrapolated to the exit. $ibcex$ determines how the pressure $prat$ is determined. = 1 $prat$ is specified as a constant. Only applicable to linear geometries or annular with zero swirl or radial outflow. = 2 Supersonic outflow. p is extrapolated to the boundary. = 3 $prat$ is specified at the hub exit. The spanwise variation of \bar{p} is found by solving radial equilibrium. \bar{p} is constant blade-to-blade. <default> = 4 $prat$ is specified at the hub exit. The spanwise variation of \bar{p} is found by solving radial equilibrium. p is found as a perturbation about \bar{p} using a characteristic boundary condition developed by Giles. = 5 For linear geometries only, \bar{p} is set at all spanwise locations, and p is found as a perturbation about \bar{p} using a characteristic boundary condition developed by Giles.
isymt	Top-plane symmetry flag. Used to model the bottom half of a linear cascade with bottom-to-top symmetry. = 1 Symmetry condition on $k = km$.

else Solid wall boundary condition on $k = km$. <default>

kbcorder Flag for order of accuracy used in endwall boundary conditions. Typically 2 (second order), but it is sometimes necessary to use 1 (first order) if points decouple spanwise. This usually only happens on linear geometries with unstretched spanwise grids. <2>

ires Iteration increment for writing residuals in the output file. Typically 10. If the code is blowing up, set $ires = 1$ to print the size and location of the maximum residual at each iteration

icrnt Iteration increment for updating the time step. Typically 50. If $icrnt$ is very large, the residual history may be discontinuous where Δt is recalculated, especially at restarts. <50>

iresti Read input restart file flag. Restart files are in PLOT3D format in the relative frame.
 = 1 Read restart file from unit 2.
 else No action taken. <default>

iresto Write output restart file flag.
 = 1 Write restart file to unit 3. <default>
 else No action taken.

iqin Inlet condition flag.
 = 0 Inlet conditions are calculated by *subroutine qincalc* based on input values $emxx$, $emty$, $dblh$, etc. The code calculates inlet P_0 and T_0 profiles using the current input values, so the inlet profiles can be changed at restart if desired.
 = 1 Inlet q-file read from unit 4. Used mainly to read an exit profile from a solution of an upstream blade row. The file is read as follows:

```

      read(4,*)kin
      do 10 k=1,kin
10    read(4,*)dum,r(k),(qin(l,k),l=1,5)

```

Here kin is the number of spanwise points, dum is a dummy variable, $r(k)$ is the radius, and $qin(l,k)$ are the five nondimensional conservation variables at that radius as described in “Solution Q-File” pp. 13, except that the absolute velocities are stored.

mioe Mass flow output flag for residual history. For transonic fans the inflow may respond slowly to a change in back pressure, so the inlet mass flow can be monitored for convergence. For turbines the inflow may choke quickly so the outflow can be monitored. In general the mass flow error is a good measure of convergence and accuracy and should converge to a fraction of a percent (e. g., < 0.003).

= 1 Inlet mass flow history is written.
 = 2 Exit mass flow history is written.
 = 3 Mass flow error $1 - \dot{m}_{out}/\dot{m}_{in}$ is written. <default>

&nam4 - Flow Parameters

igeom Linear cascade or annular blade row flag.
 = 0 Linear cascade.
 = 1 Annular blade row <default.>

ga Ratio of specific heats γ . <1.4 for air>

om Normalized blade row rotational speed, Ω/c_0 , where Ω is the wheel speed in radians per second, and c_0 has dimensions of [grid units/sec], giving om dimensions of [1/grid units]. The (x,y,x) coordinate system is right-handed, and Ω is positive in the positive x -direction. Ω is negative for most Lewis geometries. <0.>

prat	Ratio of the hub exit static pressure to the reference total pressure, $prat = p_{\text{hub exit}}/P_0$.
emxx	Nominal inlet midspan Mach number in the x or axial direction. Used for initial conditions, but u may vary with the solution for subsonic inflow.
emty	Nominal inlet midspan absolute Mach number in the y or θ direction. The sign on $emty$ must be consistent with the sign on om . Used for initial conditions. Inlet values of v or v_θ will stay constant. <0.>
emrz	Nominal inlet midspan Mach number in the z or radial direction. Used for initial conditions. Inlet values of w or v_r may vary with the solution for subsonic inflow. <0.>
expt	Exponent used to specify the inlet whirl distribution:

$$M_\theta = emty(r/r_{\text{mid}})^{\text{expt}}$$

= 0 gives uniform M_θ except within the endwall boundary layer. <default>

= -1 gives free vortex inflow.

= 1 gives forced vortex inflow.

alex Exit absolute flow angle [deg]. Used for initial conditions only. <0.>

&nam5 - Viscous Parameters

ilt	Inviscid, Laminar, or Turbulent analysis. = 0 Inviscid. The remaining viscous parameters are not used if $ilt=0$. = 1 Laminar. = 2 Turbulent using the Baldwin-Lomax turbulence model. <default> = 3 Turbulent using the Cebeci-Smith turbulence model. This model works well for turbine heat transfer (ref. 3) but may overpredict losses for transonic compressors.
renr	Reynolds number per unit length based on reference conditions, $renr = \rho_0 c_0 / \mu_0$. Must have units of [1/grid units]. Generally much larger than a conventional "free-stream" Reynolds number. For example, for standard conditions:

$$\begin{aligned} renr &= 0.002376 \left(\frac{\text{lb}_f \text{sec}^2}{\text{ft}^4} \right) \times 1116.7 \left(\frac{\text{ft}}{\text{sec}} \right) / 3.99 \times 10^{-7} \left(\frac{\text{lb}_f \text{sec}^2}{\text{ft}^2} \right) \\ &= 6.65 \times 10^6 / \text{ft} \end{aligned}$$

prnr	Prandtl number. <0.7 for air>
tw	Normalized wall temperature, $tw = T_{\text{wall}}/T_0$. = 0 Adiabatic wall boundary conditions are used. = 1 $T_{\text{wall}} = T_0$. else $T_{\text{wall}} = tw$.

vispwr Exponent for laminar viscosity power law. <default = 0.667 for air>

$$\mu/\mu_0 = (t/t_0)^{\text{vispwr}}$$

ptr	Turbulent Prandtl number. <0.9>
cmutm	Value of $\mu_{\text{turb}}/\mu_{\text{lam}}$ at which transition is assumed to occur. Baldwin and Lomax recommend 14. Can be increased to move transition downstream or vice-versa. If $cmutm = 0$, the flow is fully turbulent. <14.>
hrough	Surface roughness height in turbulent wall units h^+ . Implemented in both the Baldwin-Lomax model ($ilt=2$) and the Cebeci-Smith model ($ilt=3$) using the Cebeci-Chang roughness model. $hrough \leq 4$ gives a hydraulically-smooth surface.
jedge	j-index where the artificial viscosity begins to ramp off near the blade. Also the last j-index searched for the blade turbulent length scale. For the Baldwin-Lomax turbulence model ($ilt = 2$), <i>jedge</i> should be a grid line slightly bigger than the largest expected blade boundary layer. For the Cebeci-Smith turbulence model ($ilt = 3$), <i>jedge</i> should be a grid line slightly bigger than half the largest expected blade boundary layer. <10>
kedgh, kedgt	k-indices where the artificial viscosity begins to ramp off near the hub and tip. Also the last k-indices searched for the hub and tip turbulent length scales. See comments for <i>jedge</i> . <10>
iltin	Flag controlling inlet T_0 and P_0 profiles. = 0 Inviscid. = 1 Laminar. = 2 Turbulent using Cole's wall-wake profile. <default>
dblh, dblht	Inlet hub and tip boundary layer thicknesses in grid units.
srtp	Stationary or Rotating tip. = 0 Stationary. <default> = 1 Rotating.
xrle, xrte	Axial locations at which the hub starts and stops rotating, for modeling a rotating blade disk. Rotational boundary conditions are applied on the hub for $xrle < x < xrte$. Stationary conditions are applied elsewhere. Set $xrle < x_{\text{inlet}}$ and $xrte > x_{\text{exit}}$ to make the entire hub rotate. Note that <i>xrle</i> and <i>xrte</i> may not be sufficient to locate the rotating part of the disk in a radial flow machine.

&nam6 - Output Control

nko	Number of k-indices for blade surface output, max = 50. <0>
ko	Array of <i>nko</i> k-indices separated by commas where blade surface output is desired.

RVC3D Output

Printed output from RVC3D is written to Fortran unit 6 (standard output.) The output is divided into several sections. The sections are commonly separated using an editor and plotted using any x-y plotting package that can read ascii column data.

1. The input variables are echoed back for reference, and any comments or warnings regarding the input are given.
2. Spanwise profiles of θ – averaged flow variables are given at the inlet or exit. These variables are either based on the initial guess or on a restart file, depending on how the code is started. The initial profiles are often useful for identifying grid lines near endwall boundary layers.
3. A convergence history gives maximum and RMS residuals of density, and exit flow properties versus iteration.
4. Spanwise profiles of θ – averaged flow variables are repeated at the inlet and exit for the new solution. An approximate energy-averaging scheme is used. It gives a local representation of the average flow, not a mixed-out average.
5. Blade surface profiles of various quantities are given on selected k grid lines (spanwise locations.) Values of y^+ for the first grid point are given for checking turbulent grid spacing, and maximum values of μ_T are given to identify transition points.

Appendix - File Descriptions

Grid XYZ-File

Grids are stored using standard PLOT3D xyz-file structure. Grids can be read with the following Fortran code:

```
c      read grid coordinates
      read(1) im, jm, km
      read(1) (( (x(i, j, k), i=1, im), j=1, jm), k=1, km),
&          (( (y(i, j, k), i=1, im), j=1, jm), k=1, km),
&          (( (z(i, j, k), i=1, im), j=1, jm), k=1, km)
```

Solution Q-File

Solution files are stored in standard PLOT3D q-file structure. Solution files can be read with the following Fortran code:

```
c      read q-file
      read(2) im, jm, km
      read(2) emin, aldeg, renr, time
      read(2) ((( (qq(l, i, j, k), i=1, im), j=1, jm), k=1, km), l=1, 5)

c      additional geometry data and residual history
      read(2) itl, iil, phdeg, ga, om, nres, igeom, dum, dum, dum
      read(2) ((resd(n, l), n=1, nres), l=1, 5)
```

For linear geometries the q-variables are:

$$q_l = [\rho, \rho u, \rho v, \rho w, e]$$
$$e = \rho \left(C_v T + \frac{1}{2} \rho (u^2 + v^2 + w^2) \right)$$

For annular geometries the relative velocities are stored:

$$q_l = [\rho, \rho u, \rho v_\theta', \rho v_r', e]$$
$$e = \rho \left(C_v T + \frac{1}{2} \rho (u^2 + v_\theta'^2 + v_r'^2) \right)$$

Non-Dimensionalization

The grid xyz-file may be input in arbitrary units of length. The input parameters to RVC3D and the output q-file are strictly nondimensional, with the exception of lengths which must be input in the same units as the grid.

All quantities are nondimensionalized by an arbitrary reference stagnation state defined by total density ρ_0 and total sonic velocity c_0 . A reference viscosity μ_0 is defined by the stagnation temperature $T_0 = c_0^2 / (\gamma R)$. Standard conditions are often used for the reference state, but *any* self-consistent state may be used as long as the units of length are consistent with the grid units. The reference state is *not necessarily* the inlet state.

For input and output, pressures and temperatures are nondimensionalized by P_0 and T_0 . Within the code, pressures are usually nondimensionalized by $\rho_0 c_0^2 = \gamma P_0$.

Four input parameters must be properly nondimensionalized: *renr*, *om*, *prat*, and *tw* (see "RVC3D Input", pp. 7.)

References

1. Chima, R. V., Yokota, J. W., "Numerical Analysis of Three- Dimensional Viscous Flows in Turbomachinery," AIAA J., Vol. 28, No. 5, May 1990, pp. 798-806.
2. Chima, R. V., "Viscous Three-Dimensional Calculations of Transonic Fan Performance," in CFD Techniques for Propulsion Applications, AGARD Conference Proceedings No. CP-510, AGARD, Neuilly-Sur-Seine, France, Feb. 1992, pp 21-1 to 21-19. Also NASA TM-103800.
3. Chima, R. V., Giel, P. W., and Boyle, R. J., "An Algebraic Turbulence Model for Three-Dimensional Viscous Flows," in *Engineering Turbulence Modeling and Experiments 2*, Rodi, W. and Martelli, F. editors, Elsevier pub. N. Y., 1993, pp. 775-784. Also NASA TM-105931.
4. Baldwin, B. S., and Lomax, H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257, Jan. 1978.
5. Turkel, E., "A Review of Preconditioning Methods for Fluid Dynamics," *Applied Numerical Mathematics*, Vol. 12, 1993, pp. 257-284.
6. Chima, R. V., "TCGRID 3-D Grid Generator for Turbomachinery - User's Manual and Documentation," Oct. 1996, available from the author.
7. Sorenson, R. L., "A Computer Program to Generate Two- Dimensional Grids About Airfoils and Other Shapes by Use of Poisson's Equation," NASA TM-81198, 1980.
8. Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259, June 1981

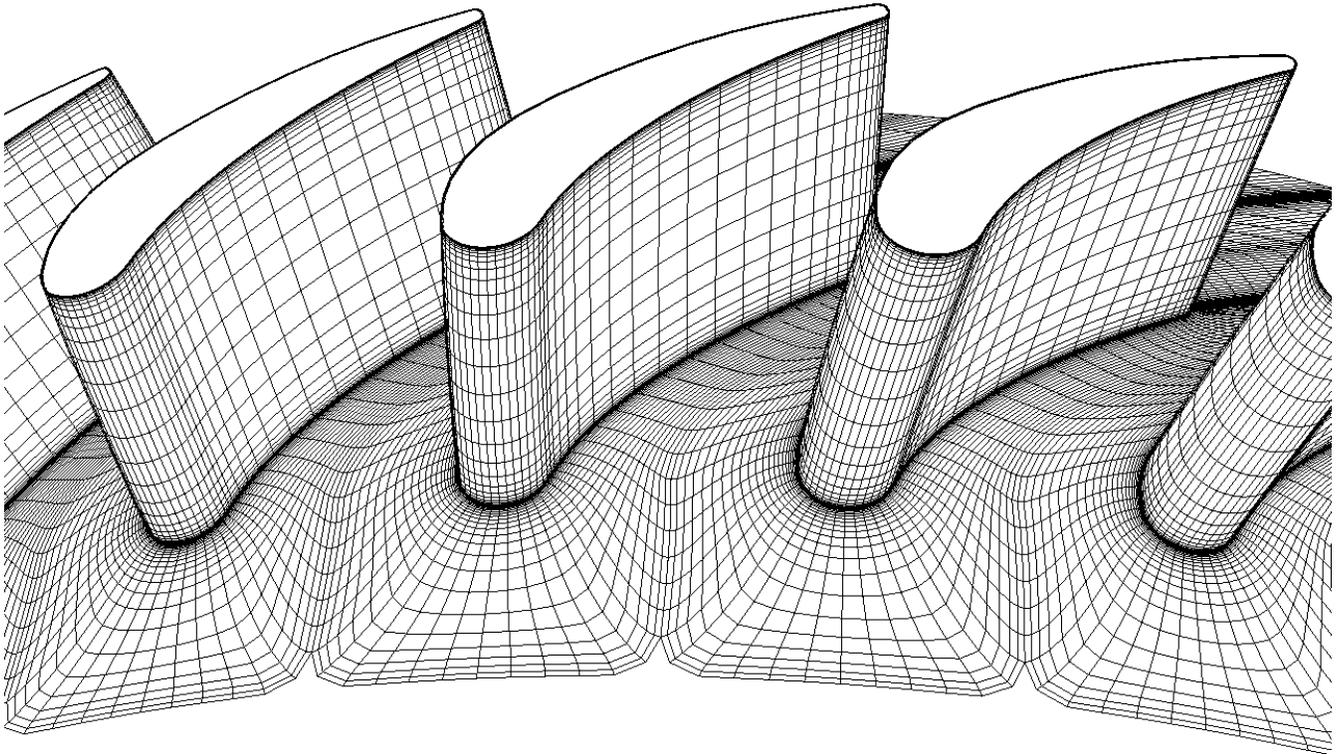


Figure 1 —Three-dimensional C-grid for a turbine vane.

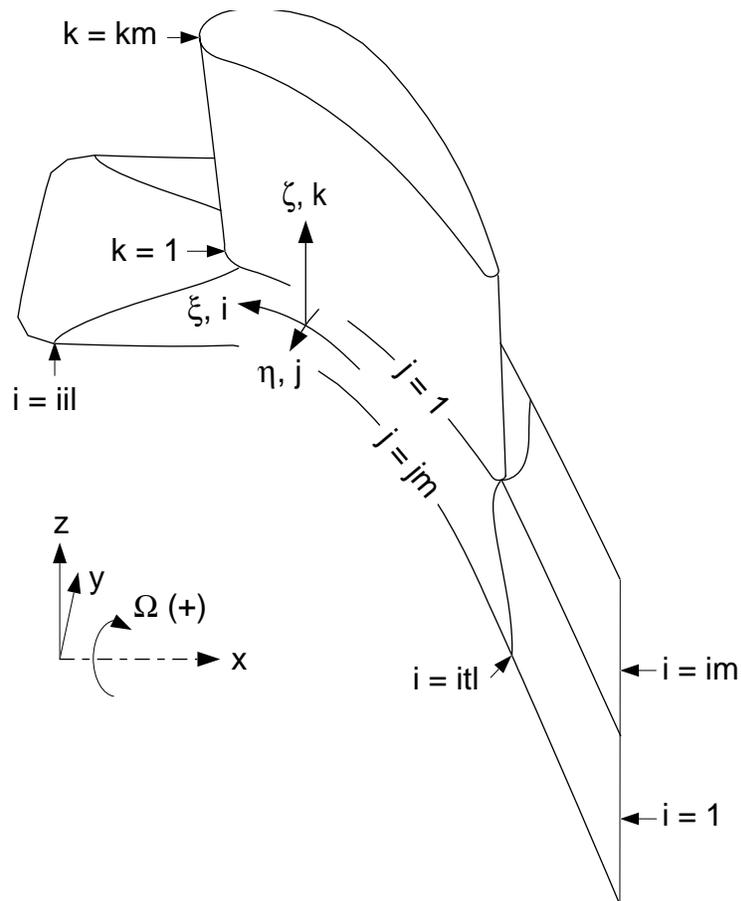


Figure 2 —Body-fitted coordinate system and index conventions for a turbine vane grid.