

# TCGRID 3-D Grid Generator for Turbomachinery

## User's Manual and Documentation

Version 206, Dec., 1999

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>

## Introduction

TCGRID (Turbomachinery C-GRID) is a three-dimensional grid generation code for turbomachinery blades. The code can generate single block grids that are compatible with the RVC3D code and multiblock grids that are compatible with the SWIFT code. Single-block grids can be either C-type or H-type, and can be for linear cascades or annular blade rows. Multiblock grids must use a C-type grid around the blade, and can add an H-grid in the inlet region and O-grids in the hub or tip clearance regions.

A brief description of TCGRID and an example of a compressor grid is given in (1). Examples of turbine grids generated with TCGRID can be found in (2). Figures 1–3 here show examples of a multiblock grid for a transonic compressor rotor (NASA rotor 37) and are used to describe the input variables later.

All geometry manipulation in TCGRID is done using parametric cubic splines, so the code can handle axial, mixed, and centrifugal flow machines. The input blade geometry can be translated, rescaled, and flipped tangentially, and full control of spacing along the blade surface is provided. Blade-to-blade grids are generated using an efficient elliptic solver that gives control of spacing and angles at the blade and outer periodic boundary. Grids are reclustered spanwise with control over spacing at the hub, casing, and clearance regions.

TCGRID is written completely in FORTRAN and runs as a quick batch job on most workstations or mainframe computers. It has been run on a PC. Code input is supplied as an ASCII dataset, with grid parameters specified as namelist input. Hub and tip geometry are specified as coordinate pairs, as are the inlet and exit boundary locations. Blade shapes may be specified in MERIDL format, Crouse-Tweedt design code format, or specified directly by the user as coordinate triplets. Some printed output is provided. No graphical output is provided, but grid 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 TCGRID code works. Instructions for dimensioning, compiling, and running TCGRID are given for Silicon Graphics (SGI) workstations and Cray mainframes. The namelist input variables and the hub, tip, and blade input are described in detail. Finally, an outline of the code structure and the output file format is given.

## How TCGRID Works

TCGRID is based on an old, two-dimensional version of the Steger/Sorenson GRAPE code (5, 6). The GRAPE code is used to generate blade-to-blade grids on several surfaces of revolution cutting through the three-dimensional blade. The individual blade-to-blade grids are stacked and reclustered spanwise to form the 3-D grid. Additional inlet or clearance blocks are then generated algebraically. An outline of the code structure and output files is given later in the section entitled *TCGRID Code Structure*. Details of the code are discussed below.

Hub and casing geometries are input as arrays of (z,r) coordinates. Blade geometries are input as arrays of (z,r,θ) coordinates. Several blade input formats are supported.

The surfaces of revolution are defined by a coarse, two-dimensional meridional grid generated algebraically by connecting corresponding points on the hub and casing. The number of spanwise points should be about the same as the number of input blade sections. The meridional grid is usually equally-spaced spanwise to give a smooth spline fit when reclustered later. However; for blades with fillets at the hub or casing, the meridional grid can be generated with approximately the same spanwise clustering as the input blade sections.

The input blade sections may overlap or be inside the hub and casing geometry, and will not generally lie along the meridional grid. For that reason the input blade sections are intersected with the meridional grid. This is done by finding the intersection of each streamwise grid line of the meridional grid with each spanwise line connecting corresponding points of each blade section. The intersection routine is iterative. It uses parametric spline fits of the coordinates and converges very rapidly unless there is a serious problem with the input.

After the blade has been intersected with the meridional grid, the individual sections are reclustered parametrically by arc length around the section. The trailing and leading edges are evenly-spaced. The trailing-edge spacing is centered on the first input point on the section (which must be repeated as the last input point for some input options.) The leading-edge spacing is centered at some fraction of arc length using the input parameter *dsra*. If *dsra*=0, the leading-edge spacing is centered on the median input point. The blade surfaces are reclustered between the leading and trailing edges using either Hermite polynomials or hyperbolic tangent clustering.

The blade-to-blade grids are generated next using the GRAPE code. The GRAPE code is strictly two-dimensional, so the three-dimensional blade sections must be mapped to a two-dimensional ( $m, \bar{r} \times \theta$ ) coordinate system. Here  $m$  is a meridional coordinate defined by  $dm^2 = dz^2 + dr^2$ .  $m$  is calculated and stored when the blades are intersected with the meridional grid.  $\bar{r}$  is simply the mean radius.

The GRAPE code requires the specification of an inner and outer boundary. For a C-grid the mapped blade coordinates plus a linear wake cut define the inner boundary. A periodic boundary is defined using the mean camber line inside the passage, a quadratic extension that turns to axial upstream, and a linear extension downstream. The outer boundary is formed by shifting the periodic boundary a half pitch up and down, and connecting the two segments at the inlet. For an H-grid the blade is bisected and extended linearly in each direction. The upper half of the blade becomes the inner boundary, and the outer boundary is formed by shifting the lower half of the blade upward one pitch.

An initial grid is generated by connecting corresponding points on the inner and outer boundaries. Angles and spacings are specified on the boundaries. The angles are set to give normal grid lines at the blade and inlet, and vertical grid lines at the periodic boundaries. The spacings are input. The GRAPE code uses an SLOR scheme to solve the Poisson equations, and is typically iterated 100 iterations. The smoothed grid is then transformed back to cylindrical coordinates.

After blade-to-blade grids have been generated on each meridional surface, they are connected spanwise and reclustered parametrically by arc length to give the full three-dimensional grid. Separate clustering functions are used in the hub clearance, blade, and tip clearance regions.

An inlet block may then be generated. The meridional boundaries are defined by the hub and casing boundaries and the H-grid and C-grid inlet boundaries. Transfinite interpolation (7) is used to fill in the interior points. The resulting meridional grid is swept tangentially so that it matches the existing points along the inlet of the C-grid.

Hub and tip clearance grid may also be generated. The outer boundary and spacing are defined by the first two C-grid lines around the blade in the clearance region. Hermite polynomials are used to connect corresponding points across the gap and to add radial spokes at the leading and trailing edges.

Finally the grid is transformed to Cartesian coordinates and output in PLOT3D multiblock format.

# Compiling and Running TCGRID

TCGRID 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 > tcgrid.f << `/\eof'
#TCGRID source code goes here
`/\eof'
cat > param << /\eof
#blade C- or H-grid size parameters
parameter(ni=319,nj=46,nk=63,nb=2*ni)
/\eof
#etc.
cat > blk01 << /\eof
#labeled common blocks
common/blk01/...
/\eof
#etc.
#compiler commands with options go here
/bin/rm -f ...
```

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

TCGRID has been set up with parameter statements to make redimensioning simple. The parameter statements and labeled common blocks are inserted during compilation using Fortran include statements, then deleted. TCGRID checks the user input against the dimensioning parameters and stops with a fatal error message if the code is not dimensioned properly. The parameter statements are given below. Typical values are shown, but they may be different in the distribution code.

*param* - Dimensions of the main C- or H-grid around the blade.

parameter(ni=319,nj=46,nk=63,nb=2\*ni)

Must have  $ni \geq im$ ,  $nj \geq km$ , and  $nk \geq km$ , where the main grid size is  $(im, jm, km)$ .  $nb$  is the number of points around the blade on an H-grid.

*para2* - Dimensions of the 2-D meridional grid and MERIDL blade input arrays.

parameter(mi=50,mm=50)

Must have  $mi \geq i2d$  and  $mm \geq \max(nbs, k2d)$ , where the 2-D meridional grid size is  $(i2d, k2d)$ , and  $nbs$  is the number of input blade sections if MERIDL blade input is used ( $merid = 2$  or  $3$ ).

*para3* - Dimensions of the hub or tip clearance O-grid block.

parameter(idt=199,jdt=13,kdt=13)

Must have  $idt \geq im - 2(itl - 1)$ ,  $jdt \geq \max(jmh, jmt)$ , and  $kdt \geq \max(kmh, kmt)$ , where the hub grid size is  $(idt, jmh, kmh)$  and the tip grid size is  $(idt, jmt, kmt)$ .

Use  $idt = jdt = kdt = 1$  if not making a hub or tip clearance grid ( $igclh = igcft = 0$  <default>).

*para4* - Dimensions of the inlet H-grid block.

parameter(idi=64,jdi=35,kdi=63)

Must have  $idi \geq imi$ ,  $jdi \geq jmi$ , and  $kdi \geq kmi$ , where the inlet grid size is  $(idi, jmi, kmi)$ .

Use  $idi = jdi = kdi = 1$  if not making an inlet grid ( $igin = 0$  <default>).

### Compiling TCGRID on Silicon Graphics Workstations (IRIX)

The following command will compile TCGRID on SGI power series or Indigo 2 workstations:

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

#SGI Indigo 2 compiler
f77 -O2 -sopt -mips2 -lfpe -o tcgrid tcgrid.f
strip tcgrid
```

### Compiling TCGRID on Cray Research Computers (UNICOS)

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

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

The `-es` option produces a compiler listing, and the `-M367` option omits the *include* statements from the listing.

### Running TCGRID

The executable program is run as a standard unix process:

```
tcgrid < std_input > std_output &
```

### Output Files

The output grid file is written to Fortran unit 1 (fort.1). Debug files are written to fort.11 - fort.19. All grid files are written as unformatted binary files. They may be linked to file names before running TCGRID,

```
ln file.xyz fort.1
```

or renamed after running TCGRID,

```
mv fort.1 file.xyz
```

Binary grid files can be used immediately by RVC3D or SWIFT 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 the solvers. Files written on a Cray while assigned as ieee binary can be used directly by RVC3D or SWIFT 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.11
#etc.
```

# TCGRID Input

TCGRID input consists of five blocks of namelist input (&nam1 - &nam5), followed by a title, then ASCII hub, tip, and blade coordinates. All grids require the title, hub, tip, and blade coordinate input, (see *Hub, Tip, and Blade Input, page 11*). Many of the namelist variables are initialized by a block data subroutine. Required input variables that are not initialized are listed below, followed by some comments regarding optional variables. All variables are described in detail in the section entitled *Namelist Input*.

## C-grids

To generate the main C-grid around the blade the following variables are required:

&nam1 - *im, jm, km, merid, itl, icap*.

To change clustering along the blade surface, along the span, and upstream and downstream of the blade, use *iclus, icluss,* and *iclusw*, respectively.

For blades with fillets use *iclus2d* to cluster the meridional grid using the same spanwise clustering as the input blade section.

For complex flow paths change the meridional grid size using *i2d* and *k2d*.

&nam2 - *nle, nte, dsle, dste, dswte, dswex, dsmin, dsmax, dshub, dstip*.

Vary the leading- and trailing-edge radii with span using *dsthr*.

Move the location of the leading-edge clustering using *dsra*.

&nam3 - *iterm*.

For a quick check of a new grid, set *iterm=0*. Use PLOT3D to check leading- and trailing-edge spacings, surface clusterings, boundary locations, and outer boundary spacings.

If something goes wrong, use the array *idbg(9)* to generate debug grids to check input coordinates or intermediate grids (see *TCGRID Code Structure, page 14*.)

Use *aabb* and *ccdd* to move points towards or away from the blade and outer boundary respectively.

&nam4 - Inlet and exit boundary coordinates *zbc* and *rbc* are required.

&nam5 - Most variables can be defaulted.

Use *zscale, tscale, rscale, ztrans,* and *tflip* to manipulate the input blade coordinates.

Use *ioble, exl,* and *exr* to change the outer boundary shape.

Use *iwakex* and *iwakex* to stretch the wake grid.

## H-grids

Most of the parameters required for C-grids are also required for H-grids. In addition, the following indices must be set:

&nam1 - *igch=1, imi, itl*.

&nam2 - Spacing parameters are interpreted as follows:

*dsin* = spacing at inlet,

*dswex* = spacing at exit,

*dsle* = spacing away from leading edge,

*dste* = spacing away from trailing edge,

*dsmin* = spacing at lower blade surface,

*dsmax* = spacing at upper blade surface.

&nam3 - Algorithm control parameters are interpreted as follows:

*aabb* and *omegpq* refer to the lower blade surface,

*ccdd* and *omegrs* refer to the upper blade surface.

## Linear Cascades

To generate a grid for a linear cascade set the following:

&nam1 - *igeom = 1,*

&nam2 - *gap*.

## Inlet H-grid

To generate an inlet H-grid, set

&nam1 - *igin = 1, imi*.

&nam2 - *dsin*.

Since the inlet H-grid overlaps the blade C-grid, the C-grid must be run to convergence for proper H-grid spacings.

&nam5 - *iswift=1*

### Tip Clearance O-grid

To generate a tip clearance grid, set the following:

&nam1 - *igclt* = 1, *jmt*, *kmt*.

&nam2 - *cltip*, *dsclt*.

The number of points in the i-direction depends on the C-grid size. Since the clearance O-grid overlaps the blade C-grid, the C-grid must be run to convergence for proper O-grid spacings.

&nam5 - *iswift*=1

### Hub Clearance O-grid

To generate a hub clearance grid, set the following:

&nam1 - *igclh* = 1, *jmh*, *kmh*.

&nam2 - *clhub*, *dsclh*.

The number of points in the i-direction depends on the C-grid size. Since the clearance O-grid overlaps the blade C-grid, the C-grid must be run to convergence for proper O-grid spacings.

&nam5 - *iswift*=1

## Namelist Input

Defaults are given in angle brackets, <Default=value> or <default.> If no default is given the value MUST be input. Relevant figures are given in parentheses (fig. #.)

### &nam1 - Grid Size Parameters

Many grid size parameters are illustrated in figures 1–3.

<i>merid</i>	Flag for type of blade input. See <i>Hub, Tip, and Blade Input</i> , page 11, and figure 4 for complete descriptions of the blade input formats.  = 0 Blade input in stacked sections, (z, r, $\theta$ ). Completely general, (fig. 4), <default.>  = 1 Blade input in Crouse/Tweedt design code format, (z, r, $\theta$ ). Similar to above but ordered differently.  = 2 Blade input in MERIDL format, (z, r, $\theta$ -upper, $\theta$ -lower), (fig. 5).  = 3 Blade input in MERIDL format, (z, r, $\theta$ , $\Delta\theta$ .)
<i>im</i>	Grid size in i- (streamwise) direction, (fig. 1).
<i>jm</i>	Grid size in j- (blade-to-blade) direction, (fig 1).
<i>km</i>	Grid size in k- (spanwise) direction, (fig. 3).
<i>itl</i>	i-index of lower trailing-edge point on a C-grid, (fig. 1). Trailing-edge index for an H-grid.
<i>icap</i>	Number of i-points on the inlet part of the C-grid, equally-spaced. Remaining points are distributed over the periodic boundaries. Increase <i>icap</i> to pull points towards inlet, and vice-versa, (fig. 1).
<i>igeom</i>	Flag that tells whether grid will be for a linear cascade or an annular blade row.  = 0 Annular blade row <default.>  = 1 Linear cascade.
<i>iclus</i>	Flag for type of clustering along the blade surfaces.  = 1 Hyperbolic tangent clustering - smoothest, but may be sparse at blade center if <i>im</i> is small <default.>  = 2 Hermite polynomial clustering - more uniform, but may grow too quickly near leading and trailing edges. Good for large <i>im</i> .
<i>icluss</i>	Same as <i>iclus</i> , but for clustering in the spanwise direction.
<i>iclusw</i>	Same as <i>iclus</i> , but for streamwise clustering downstream in the wake, and also upstream for an H-grid.

<i>iclus2d</i>	Flag that sets spanwise clustering of the 2-D meridional grid. = 0 2-D grid is equally-spaced. = 1 2-D grid has approximately the same spanwise clustering as the input blade sections. Use this option if the input blade sections resolves fillets <default.>
<i>i2d</i>	Number of i- (streamwise) points on the coarse meridional grid used to define the passage. Typically 21 for an axial machine, 41 for a centrifugal. <Default = 21.>
<i>k2d</i>	Number of k- (spanwise) points on the coarse meridional grid used to define the passage. Should be roughly equal to the number of input blade sections <i>nbs</i> . <Default = 11.>
<i>igch</i>	Flag to set C- or H-grids. TCGRID can generate single-block H-grids that may be compatible with other codes. = 0 C-grid <default.> = 1 H-grid.
<i>ilh</i>	Number of i- (streamwise) points from inlet to leading edge of H-grid, (fig. 1). Only used if <i>igch</i> = 1.
<i>igin</i>	Flag to generate inlet H-grid. = 0 No inlet H-grid <default.> = 1 Generate inlet H-grid.
<i>imi</i>	Number of i- (streamwise) points in the inlet H-grid. Only used if <i>igin</i> = 1.
<i>igclh</i>	Flag to generate hub clearance O-grid. = 0 No hub clearance O-grid <default.>. = 1 Generate hub clearance O-grid.
<i>jmh</i>	Number of j- (blade thickness/2) points in the hub clearance grid. Only used if <i>igclh</i> = 1.
<i>kmh</i>	Number of k- (spanwise) points in the tip clearance grid, (fig. 3). Only used if <i>igclh</i> = 1.
<i>igclt</i>	Flag to generate tip clearance O-grid. = 0 No tip clearance O-grid <default.> = 1 Generate tip clearance O-grid.
<i>jmt</i>	Number of j- (blade thickness/2) points in the hub clearance grid. Only used if <i>igclt</i> = 1.
<i>kmt</i>	Number of k- (spanwise) points in the tip clearance grid, (fig. 3). Only used if <i>igclt</i> = 1.

### **&nam2 - Grid Spacing Parameters**

All values must be input in the units desired for the final grid. All spacing parameters named “*ds...*” refer to spacing along some arc length, and not in a particular coordinate direction. Values suggested as “e.g.” should give a good initial guess but may need to be modified after examining the initial grid.

<i>nle</i>	Number of points equally-spaced around the blade leading edge, typically 15.
<i>nle</i>	Number of points equally-spaced around the blade trailing edge, typically 10. Should be an even number.
<i>dsle</i>	Spacing around the leading edge at the hub, e.g. $\pi \cdot rle/nle$ .
<i>dste</i>	Spacing around the trailing edge at the hub, e.g. $\pi \cdot rte/nle$ .
<i>dswte</i>	Spacing away from the trailing edge on the wake cut of C-grid, should be $\approx dste$ , (fig. 1).
<i>dswex</i>	Spacing at exit on wake cut of C-grid, hard to estimate in advance. Should be roughly the spacing along the periodic boundary, which is roughly the streamwise distance from the inlet to the exit divided by $(im-icap)/2$ , (fig. 1).

<i>dsthr</i>	“ds tip-to-hub ratio.” <i>Dsle</i> , <i>dste</i> , and <i>dswte</i> are taken as hub values and are varied linearly with span to this factor at the tip. Allows the leading edge radius, etc. to increase or decrease (usually decrease) with span. <Default = 1.>
<i>dsmin</i>	Spacing away from the blade, e.g. chord/10,000 for viscous grids, (fig. 1).
<i>dsmax</i>	Spacing away from the periodic boundary, e.g. midspan-pitch/ <i>jm</i> , (fig. 1).
<i>dsin</i>	Spacing away from the inlet of inlet H-grid, (fig. 1). Only used if <i>igclt</i> = 1.
<i>dsra</i>	(Pressure surface arc length)/(total surface arc length). Used to locate the center of the leading edge clustering on the blade. The clustering is centered about <i>dsra</i> x (total surface arc length.) Typical values are 0.5 for symmetrical blades, about 0.49 for compressor blades, and about 0.45 for highly-cambered turbine blades.  = 0 TCGRID assumes that there are the same number of blade input coordinates on each surface and clusters about the median input point <default.>
<i>gap</i>	Blade row pitch for a linear cascade. Only used if <i>igeom</i> = 1. The pitch is set by <i>nblade</i> if <i>igeom</i> = 0. <Default = 1.>
<i>rcorn</i>	Radius for the front corner of the C-grid, (fig. 1). Can be 0., but the inlet is smoother with <i>rcorn</i> $\approx$ pitch/8. <Default = 0. Reset to 0. if <i>igin</i> = 1.>
<i>dshub</i>	Spanwise-spacing at the hub, e.g. span/10,000 for viscous grids, (fig. 3).
<i>dsclh</i>	Spanwise-spacing at the blade edge nearest the hub, (fig. 3). Only used if <i>clhub</i> > 0. <Default = <i>dshub</i> .>
<i>clhub</i>	Hub clearance, (fig. 3). Used if <i>igclh</i> = 1, or to cluster hub grid for a simple periodicity clearance model.  = 0 Grid is stretched continuously away from the hub <default.>  > 0 Grid is clustered near the hub using <i>dshub</i> , <i>clhub</i> , and <i>dsclh</i> .
<i>dstip</i>	Spanwise-spacing at the tip, e.g. span/10,000 for viscous grid. Should be $\approx$ <i>dshub</i> , (fig. 3).
<i>dsclt</i>	Spanwise-spacing at the edge of the blade near the tip, (fig. 3). Only used if <i>cltip</i> > 0. <Default = <i>dstip</i> .>
<i>cltip</i>	Tip clearance, (fig. 3). Used if <i>igclt</i> = 1, or to cluster tip grid for a simple periodicity clearance model.  = 0 Grid is stretched continuously away from the tip <default.>  > 0 Grid is clustered near the tip using <i>dstip</i> , <i>cltip</i> , and <i>dsclt</i> .

### **&nam3 -Algorithm Parameters**

See Sorenson’s GRAPE code documentation (5) for more information on algorithm parameters. Most values can be defaulted.

<i>iterm</i>	Number of iterations for elliptic solver, usually 50 – 150. Use <i>iterm</i> =0 to check initial grid spacings, boundary locations, etc. <default=100.>
<i>idbg(9)</i>	Integer flag array with nine elements for writing intermediate debug grids to Fortran units 11-19. Useful for debug, graphics, and possibly for grid generation in itself. For more information see <i>TCGRID Code Structure</i> , page 14. Available options are given in Table 1 on page 17. <Default = 9*0.>
<i>omega</i>	Relaxation factor for the elliptic solver. Acceptable values from 0. to 2. <Default = 1.4.>
<i>omegpq</i>	Relaxation factor for the inner boundary forcing functions. Acceptable values from 0. to 2. Set to 0. for a Laplacian inner boundary. <Default = 0.1.>
<i>omegrs</i>	Like <i>omegpq</i> , but for the outer boundary.
<i>aabb</i>	Exponent controlling the distance that angles and spacings at the inner boundary propagate into the interior. Small <i>aabb</i> give large distances but slow convergence, and vice versa. Any value > 0. is acceptable. <Default = 0.45>
<i>ccdd</i>	Like <i>aabb</i> , but for the outer boundary.

#### **&nam4 - Boundary Coordinates**

*zbc*(3,2) and *rbc*(3,2)

Arrays of (z, r) coordinates that define three line segments that act as the upstream H-grid inlet, the blade C-grid inlet, and the blade C-grid exit. The line segments are intersected with the hub and tip geometry, so the coordinates need not lie exactly on the hub and tip. Anywhere nearby should work. Figure 2 illustrates the locations of the boundary coordinate points. The first index indicates the segment and the second index indicates hub or tip. The six points must be entered in the order shown below. The coordinates of the upstream H-grid inlet may be set to zero if *igin* = 0.

*zbc* = z-H-hub-in, z-C-hub-in, z-C-hub-ex, z-H-tip-in, z-C-tip-in, z-C-tip-ex

*rbc* = r-H-hub-in, r-C-hub-in, r-C-hub-ex, r-H-tip-in, r-C-tip-in, r-C-tip-ex

#### **&nam5 - Miscellaneous Parameters**

Most of these parameters can be defaulted.

*iswift* Output file format flag.  
= 0 RVC3D code output – single block, no dummy grid lines <default.>  
= 1 SWIFT code output – one or more blocks with dummy grid lines, SWIFT index file written on unit 10.  
= 2 ADPAC code output – one or more blocks, no dummy grid lines.

#### **Scaling Parameters**

Parameters for rescaling, translating and flipping the input blade coordinates.

*zscale* Scale factor for blade z-coordinates, <default = 1.>  
*tscale* Scale factor for blade  $\theta$ -coordinates, <default = 1.>  
*rscale* Scale factor for blade r-coordinates, <default = 1.>  
*ztrans* Translation distance for blade z-coordinates, <default = 0.>  
*tflip* Flag for flipping the blade in the  $\theta$ -direction and reordering the points.  
= 0 Do not flip blade  $\theta$ -coordinates, <default.>  
= 1 Flip blade  $\theta$ -coordinates.

#### **Outer Boundary Shape Control Parameters**

*dslap* i- (streamwise) spacing at exit of C-grid. If *dslap* > 0, the grid lines at  $i = 2$  and  $i = im-1$  are repositioned exactly *dslap* from the exit, overriding *dswex*. For multistage machines the next blade row should have *dsmax* = *dslap* to give a perfect overlap of the grids.

*exl* Controls the shape of the left (upstream) periodic boundary of a C-grid. The boundary starts tangent to the mean camber line and curves to axial at a rate determined by *exl*.  
> 10 No curvature – the boundary is a linear extension of the mean camber line.  
> 1.5 Slow curvature to axial.  
= 1.5 Moderate curvature to axial <default.>  
< 1.5 Fast curvature to axial.  
= 1 Turns the boundary abruptly to axial.

*exr* Controls the shape of the right (downstream) periodic boundary of a C-grid. The boundary starts tangent to the mean camber line and curves to axial at a rate determined by *exr*.  
> 10 No curvature – the boundary is a linear extension of the mean camber line <default.>

- > 1.5 Slow curvature to axial.
- = 1.5 Moderate curvature to axial.
- < 1.5 Fast curvature to axial.
- = 1 Turns the boundary abruptly to axial.

<i>fswake</i>	Fractional distance along the downstream periodic boundary between the trailing edge and the grid exit, where the $\eta$ -grid lines from the trailing edge ( $i = itl$ ) intersect the outer boundary, (fig. 1). The default value of <1.> places the outer boundary point directly above and below the trailing edge point. On some blades this can cause the $\eta$ -grid lines to cross the trailing edge To pull the grid lines towards the downstream boundary by setting <i>fswake</i> < 1.0 .
<i>ioble</i>	The periodic outer boundary for a C-grid is made up of three segments, an upstream segment, the mean-camber line between the blades, and a downstream segment. The parameter <i>ioble</i> is an index which determines where the upstream segment joins the mean camber line. Values can be <11>, 10, 9 ... The default <11> starts the upstream segment at the leading edge. Smaller values move the starting point inside the passage, which can be useful if upstream part of the C-grid becomes distorted due to stagger, (fig. 1).
<i>iwakex</i>	Flag for stretching the outer boundary grid spacing along the wake (i-direction). = 0 Equally-spaced outer boundary along the wake. = 1 Stretched outer boundary along the wake, <default.>
<i>jwakex</i>	Flag for controlling the $\eta$ -grid spacing along the trailing edge cut. = 0 $\eta$ -grid spacing is <i>dsmn</i> all along the trailing edge cut, <default.> = 1 $\eta$ -grid spacing expands from <i>dsmn</i> at the trailing edge to equally-spaced at the exit.

### Parameters for Blunt Trailing Edges

TCGRID can now wrap C-type grids around blades with blunt trailing edges like the centrifugal impeller blade shown in figure 6b or the inlet guide vane shown in figures 6c,d. Blade coordinates may be input in one of two ways depending on the value of *merid*.

<i>merid</i>	= 0 or 1 (general coordinate input) Blades must be input with an open trailing edge (fig. 6a.) = 2 or 3 (MERIDL input) Blades are always input with an open trailing edge (fig. 5.) If <i>nbase</i> = 0 a round trailing edge is added <default.> If <i>nbase</i> > 0 a blunt trailing edge is assumed.
<i>nbase</i>	Number of intervals on the blunt trailing edge (fig. 6a.)
<i>ibase</i>	Flag controlling the location of the wake cut line with respect to the base of the blade (fig. 6.) To minimize grid distortion, choose <i>ibase</i> such that the cut leaves the corner with the acute angle. If the base is symmetric (fig. 6c,d), use <i>ibase</i> = 0. = +1 Cut line leaves the upper corner of the base. = 0 Cut line leaves the center of the base <default.> = -1 Cut line leaves the lower corner of the base.
<i>ibevel</i>	Flag for beveling the corner(s) of the base to reduce grid distortion (fig 6c,d.) If <i>ibase</i> = 0 both corners are beveled. If <i>ibase</i> = 1 the corner opposite the wake cut is beveled. = 0 No bevel <default.> = 1 Corner(s) are beveled.

# Hub, Tip, and Blade Input

Immediately following the namelist input are unformatted reads for a title, the hub and tip coordinates, and the blade coordinates. Unformatted ASCII reads are used throughout.

## Title

A title of 80 characters or less is read using the following FORTRAN input statement:

```
read *, ititle
```

*ititle* An alphanumeric string of 80 characters printed to the output. The character string must be enclosed in single quotes.

## Hub and Tip Geometry

Hub and tip coordinate arrays as shown in figure 2 are read in as follows:

```
c      read hub & tip geometry
      read(5,*) nph, npt
      read(5,*) (zhub(i), i=1, nph)
      read(5,*) (rhub(i), i=1, nph)
      read(5,*) (ztip(i), i=1, npt)
      read(5,*) (rtip(i), i=1, npt)
```

*nph* Number of input hub points, min = 2, max = ni.

*npt* Number of input tip points, min = 2, max = ni.

*zhub, rhub* z, r coordinates of the hub.

*ztip, rtip* z, r coordinates of the tip.

## Blade Geometry

The next line of input contains three variables read as follows:

```
c      blade input
      read(5,*) nbs, npb, nblade
```

where

*nbs* Number of blade sections, max. = mm.

*npb* Number of points around the blade, max. = nb.

*nblade* Number of blades around the wheel.

This is followed by blade coordinates in one of four formats determined by the input value of *merid*. The coordinates need not intersect the hub and tip coordinates – they are spline fit if they span the endwalls, or are linearly extrapolated if they do not, and the intersections are calculated by TCGRID. The four input options and their corresponding Fortran reads are as follows:

*merid* = 0 <default>

Blade input in general stacked sections. Cylindrical coordinates starting at the blade trailing edge, wrapping clockwise around the blade, and repeating the trailing-edge point. Complete definition of the leading- and trailing-edges must be given. Sections are stacked from hub to tip. Figure 4 shows the ordering of points for *merid* = 0.

```
c      merid=0: blade input in stacked sections, cyl. coords.
      if (merid.eq.0) then
      do 3 k=1, nbs
```

```

      read(5,*) (zb(i,k), i=1, npb)
      read(5,*) (yb(i,k), i=1, npb)
3     read(5,*) (rb(i,k), i=1, npb)
      endif

```

Here (zb, yb, rb) = (z,  $\theta$ , r)-coordinates of the blade section.

*merid* = 1

Blade input in Crouse/Tweedt design code format. See Tweedt's writeup on design code options. The point ordering around the blade is the same as for *merid* = 0, as shown in figure 4, but the points are stacked from tip to hub.

```

c     merid=1: blade input in Crouse/Tweedt design code format
      if(merid.eq.1) then
      read(5,*) zhub
      do 5 k=nbs,1,-1
      read(5,*) dum
      read(5,*) dum
      read(5,*) (zb(i,k), i=1, npb)
      read(5,*) (yb(i,k), i=1, npb)
5     read(5,*) (rb(i,k), i=1, npb)
      endif

```

Again (zb, yb, rb) = (z,  $\theta$ , r)-coordinates of the blade section.

*zhub*            A z-translation value added to all blade z-coordinates, to shift them to the same reference as the hub and tip. Can also be done using namelist variable *ztrans*.

*dum*             Two dummy records are included before each blade section.

*merid* = 2 or 3

Blade input in MERIDL format. See Katsanis and McNally's report on MERIDL (3) for more information on MERIDL input. Unlike MERIDL, TCGRID can handle purely radial flows without rotating the coordinate system. MERIDL input has no leading- or trailing-edge definition, but TCGRID will add leading- and trailing-edge circles automatically. Points are input from leading edge to trailing edge, and from hub to tip.

```

c     merid=2 or 3: blade input in MERIDL format
      if(merid.gt.1) then
      do 20 k=1, nbs
20    read(5,*) ( zbl(i,k), i=1, npb)
      do 22 k=1, nbs
22    read(5,*) ( rbl(i,k), i=1, npb)
      do 24 k=1, nbs
24    read(5,*) ( th1bl(i,k), i=1, npb)
      do 26 k=1, nbs
26    read(5,*) ( th2bl(i,k), i=1, npb)
      endif

```

*merid* = 2        (zbl, rbl, th1bl, th2bl) = (z, r,  $\theta$ -upper-surface,  $\theta$ -lower-surface) coordinates of the blade section, as shown in figure 5.

*merid* = 3        (zbl, rbl, th1bl) = (z, r,  $\theta$ )-coordinates of the mean-camber-line, ordered like *merid* = 2 (fig. 5),  
                   th2bl = blade tangential thickness ( $\theta$ -upper-surface –  $\theta$ -lower-surface).

## Grid Output XYZ-File

Grids are stored using standard PLOT3D xyz-file structure. Single block 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)
```

See PLOT3D documentation or the TCGRID source code for details on using multiblock grids.

# TCGRID Code Structure

Grids are generated in several sequential steps. Most steps are coded as separate subroutines, many of which can generate PLOT3D compatible grid files for debugging purposes. The outline below lists the main subroutines of TCGRID in the order that they are called, describes their function, and describes the debug flag and debug output generated. Indentation implies subroutine nesting.

Debug files are requested using the namelist array *idbg*(9) described below in Table 1. In general, if *idbg*(*n*) = 1, a debug grid file will be generated on Fortran unit *n* + 10. Grids are in PLOT3D format, may be 2-D or 3-D, and may be in multigrid format. If there is a problem with TCGRID, set *idbg* = 9\*1 and plot fort.11 – fort.19 in turn. (Some files may not be generated depending on several input options.) Refer to the outline below to determine at which step the problem occurred.

## TCGRID

Main calling program.

Output file: unit 1, 3-D grid, possibly multigrid format.

### 1. INPUT

Reads the namelist, hub, tip, and blade input. Scales and translates the geometry if desired.

Output file: *idbg*(1) = 1, unit 11, 3-D blade as input (*merid* = 0 or 1), or 3-D MERIDL blade after addition of leading- and trailing-edge circles (*merid* = 2 or 3).

### 2. INNER (*merid* = 0 or 1)

Reclusters points around the blade sections.

Adds points on the base if *nbase* > 0.

### 3. MERFIX (*merid* = 2 or 3)

Converts MERIDL blade sections to GRAPE-type sections.

Adds leading- and trailing-edge circles.

Adds points on the base if *nbase* > 0.

Reclusters points around the blade sections.

Output file: *idbg*(2) = 1, unit 12, 3-D multigrid MERIDL blade as input (*merid* = 2 or 3).

#### LETE

Computes leading- and trailing-edge circles for the MERIDL blades using the technique described in (4).

### 4. ADDHT

Adds inlet and exit points to hub and tip arrays.

### 5. MERIDG

Generates a coarse, equally-spaced meridional grid between the supplied hub and tip.

Output file: *idbg*(3) = 1, unit 13, 2-D meridional grid between hub and tip.

### 6. BLADES

Interpolates the blade geometry onto the meridional grid.

Output file: *idbg*(4) = 1, unit 14, 3-D blade after interpolation onto meridional grid.

### 7. GRID2D

Generates 2-D blade-to-blade grids along the meridional grid lines in  $(m, \bar{r} \times \theta)$  coordinates, where *m* is the meridional coordinate defined by  $dm^2 = dz^2 + dr^2$  and  $\bar{r}$  is the mean radius. Uses an old version of the Steger/Sorenson GRAPE code (refs. 5 and 6).

Output file: *idbg*(5) = *k*, unit 15, 2-D blade-to-blade grid on surface *k* of the meridional grid.

#### GRELAX

Solves the elliptic grid equations.

#### DUMGL

Adds dummy grid lines if *iswift* = 1.

Transforms the  $(m, \bar{r} \times \theta)$  coordinates to  $(z, r, \theta)$ .

Output file: *idbg*(6) = 1, unit 16, 3-D grid before spanwise clustering.

8. FILL3D  
Reclusters the 2-D grids spanwise using either Hermite polynomials or hyperbolic tangent clustering (ref. 7) to make a full 3-D grid.
9. GINLET  
Generates an algebraic H-grid block upstream of the blade using transfinite interpolation (ref. 7) if requested.  
Output file: *idbg(7) = 1*, unit 17, 3-D inlet H-grid.
10. GTIP  
Generates an algebraic O-grid block in the hub or tip-clearance region if requested.  
Output file: *idbg(8) = 1*, unit 18, 3-D tip clearance O-grid.  
Output file: *idbg(9) = 1*, unit 19, 3-D tip clearance O-grid.
11. OSPAN  
Prints the inlet, leading edge, trailing edge, and exit coordinates.
12. OUTMG  
Transforms  $(z, r, \theta)$  to  $(x, y, z)$  and writes the grid file on unit 1 in PLOT3D format.  
Output file: *iswift = 1*, unit 10, ASCII index file for use by the SWIFT code.

## Acknowledgments

Joe Steger and Reece Sorenson developed and supplied the original GRAPE code that serves as the basic grid solver in TCGRID. Kevin Kirtley developed an early version of TCGRID called GRD3D. Larry Schuman provided the algorithm for adding leading edge circles to MERIDL blades. Dan Tweedt developed the spline routines used in TCGRID and provided many helpful comments and suggestions. Dave Miller provided the transfinite interpolation routine used for the upstream grid block.

## References

1. 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.
2. Chima, R. V., Giel, P. W., and Boyle, R. J., "An Algebraic Turbulence Model for Three-Dimensional Viscous Flows," in *Engineering Turbulence Modelling and Experiments 2*, Rodi, W. and Martelli, F. editors, Elsevier pub. N. Y., 1993, pp. 775-784. Also NASA TM-105931.
3. Katsanis, T., McNally, W. D., "Revised FORTRAN Program for Calculating Velocities and Streamlines on the Hub-Shroud Midchannel Stream Surface of an Axial-, Radial-, or Mixed-Flow Turbomachine or Annular Duct," NASA TN D-8430, Mar. 1977.
4. Schumann, L. F., "FORTRAN Program for Calculating Leading-and Trailing-Edge Geometry of Turbomachine Blades," NASA TM X-73679, June, 1977.
5. 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.
6. Steger, J. L., and Sorenson, R. L. "Automatic Mesh Point Clustering Near A Boundary in Grid Generation with Elliptic Partial Differential Equations," *Journal of Computational Physics*, Vol.33, No. 3, Dec. 1979, pp.405-410.
7. Thompson, J. F., Warsi, Z. U. A., Mastin, C. W., *Numerical Grid Generation Foundations and Applications*, North-Holland, N. Y., 1985.

i	Value	Unit	Subroutine	Grid Size	Plot3d Format	Grid Description
1	1	11	Input	npb, nbs, 1	3d/unf	General blade as input ( <i>merid</i> = 0, 1) MERIDL blade after l.e. & t.e. addition ( <i>merid</i> = 2, 3)
2	1	12	Merfix	nbs, npb, 1	3d/unf/mg	MERIDL blade as input ( <i>merid</i> = 2, 3)
3	1	13	Meridg	i2d, k2d	2d/unf	2-D meridional grid between hub and tip
4	1	14	Blades	npb, k2d, 1	3d/unf	3-D blade after interpolation onto 2-D grid
5	k	15	Grid2d	im, jm, 1	2d/unf	2-D blade-to-blade grid on section k
6	1	16	Grid2d	im, km,k2d	3d/unf	3-D rid before spanwise clustering
7	1	17	Ginlet	imi, jmi, kmi	3d/unf	3-D inlet H-grid
8	1	18	Gtip	imt, jmt, kmt	3d/unf	3-D hub clearance O-grid
9	1	19	Gtip	imh,jmh,kmh	3d/unf	3-D tip clearance O-grid

Table 1 —Debug grid files available with *idbg* options.





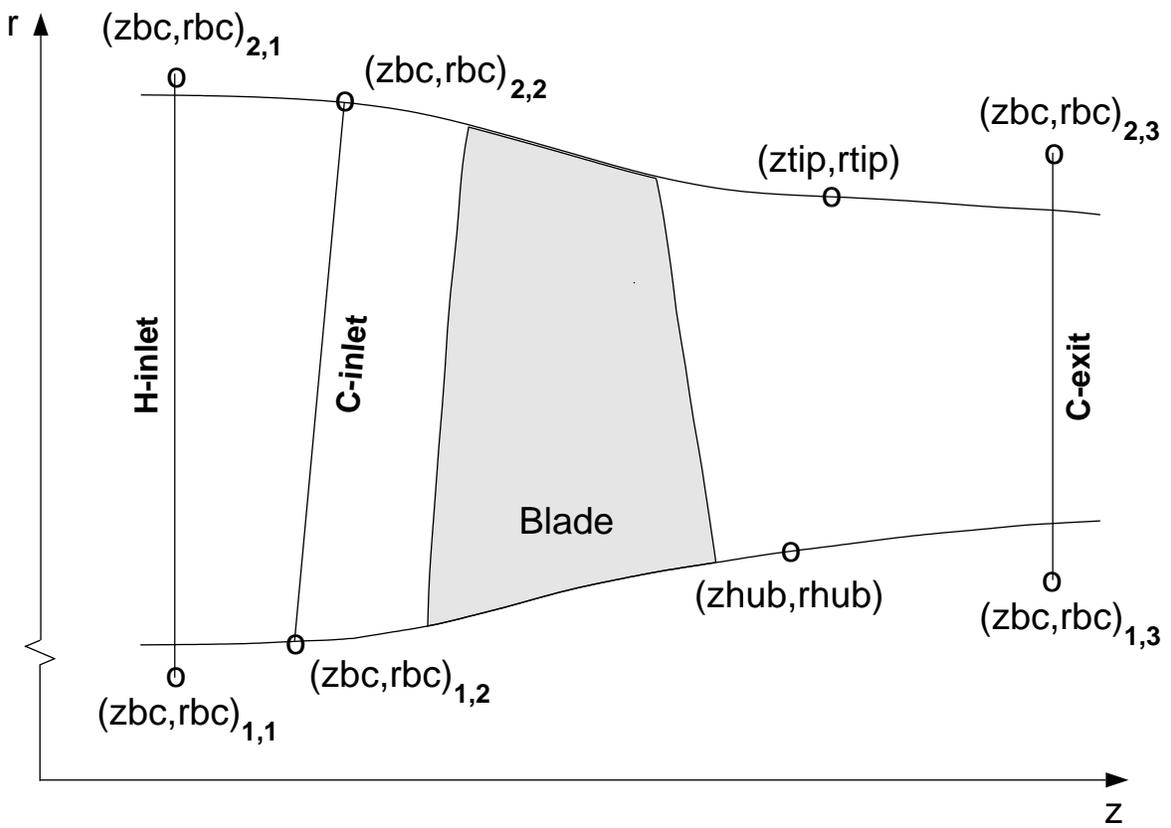
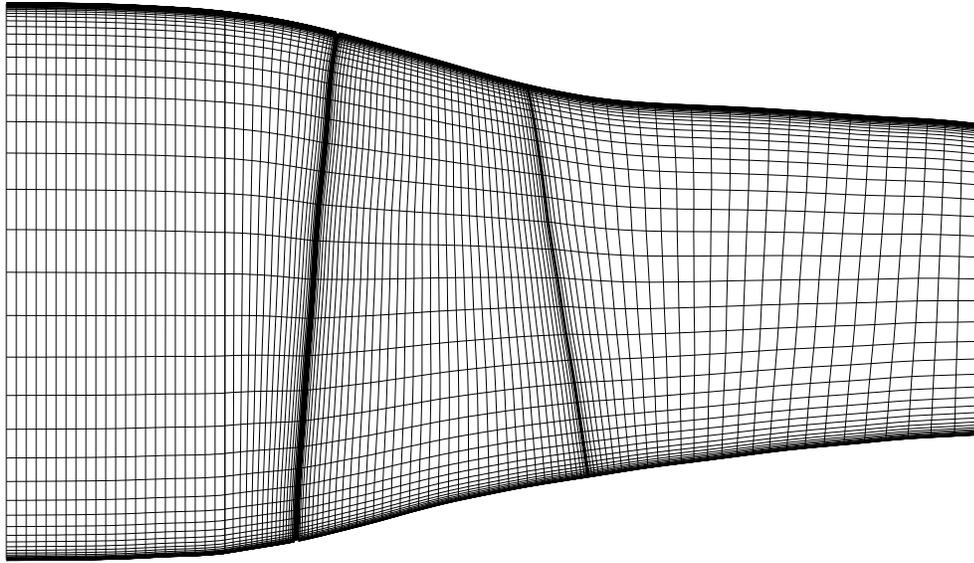


Figure 2 — (Top) Meridional H-C grid for a transonic compressor rotor.  
 (Bottom) TCGRID nomenclature and input variables for meridional grid.

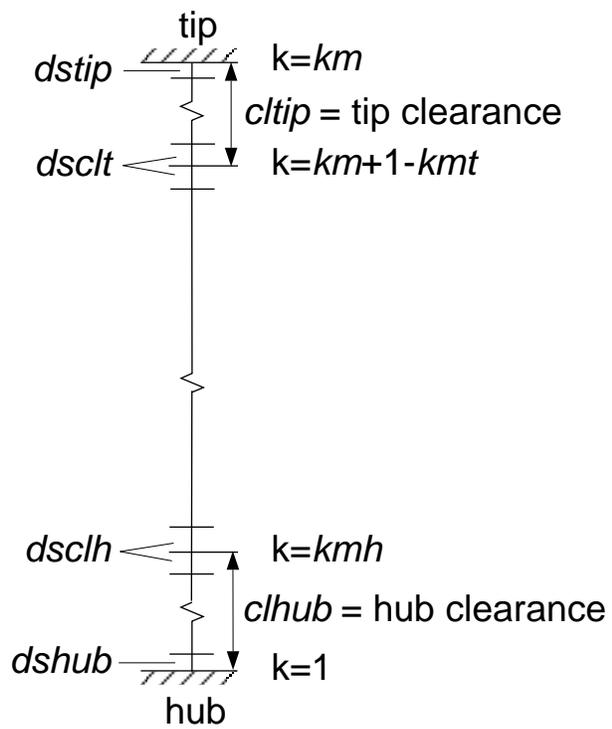
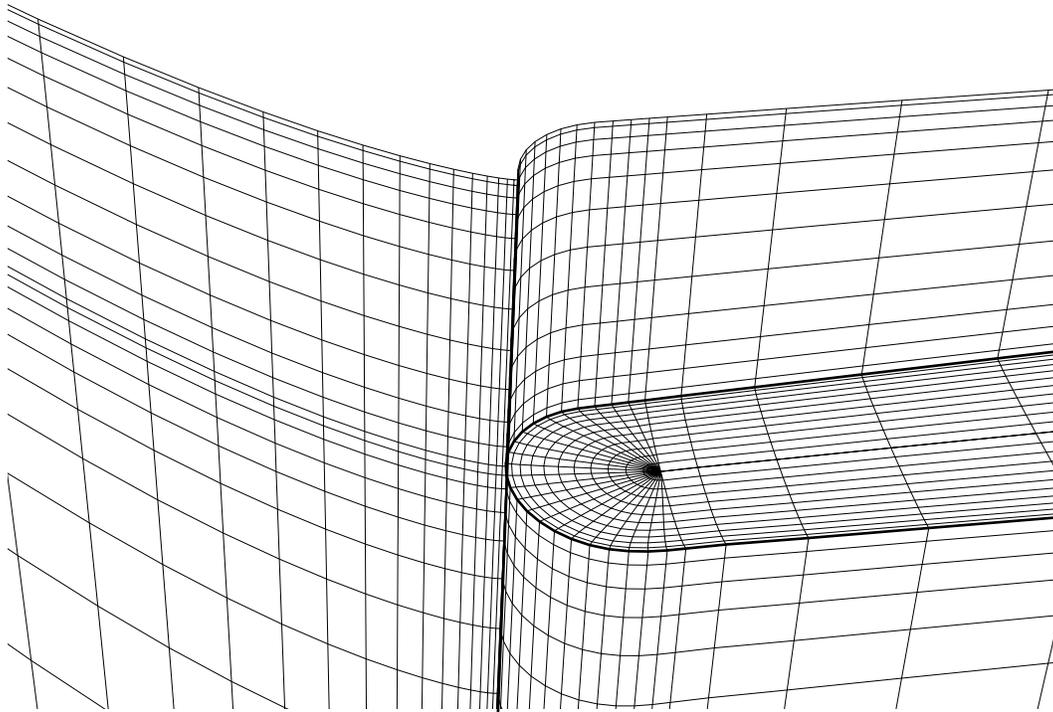


Figure 3 — (Top) Tip clearance O-grid for a transonic compressor rotor.  
 (Bottom) TCGRID nomenclature and input variables for spanwise grid.

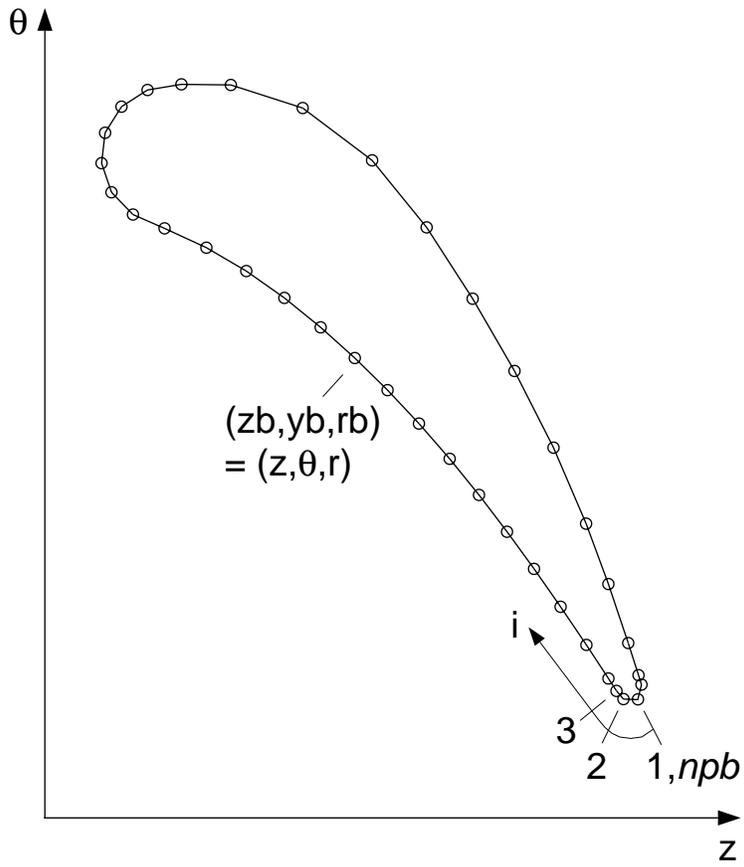


Figure 4 — Blade coordinate input variables for  $merid=0$  and 1.

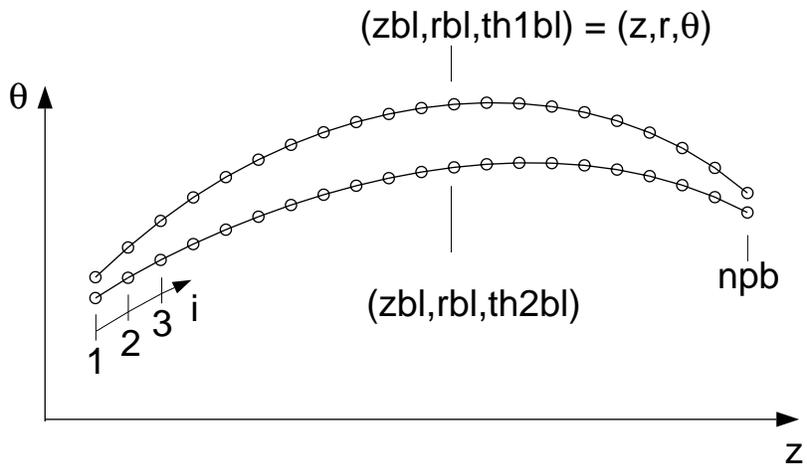


Figure 5 — Blade coordinate input variables for  $merid=2$ .

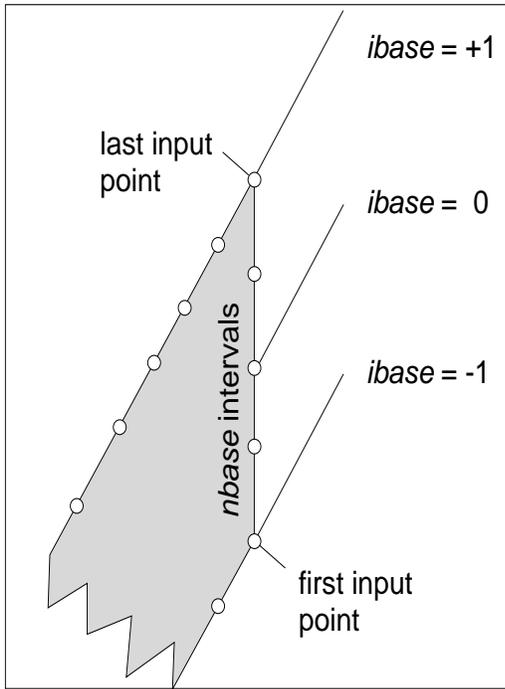


Figure 6a — Effect of  $ibase$  on location of wake cut.

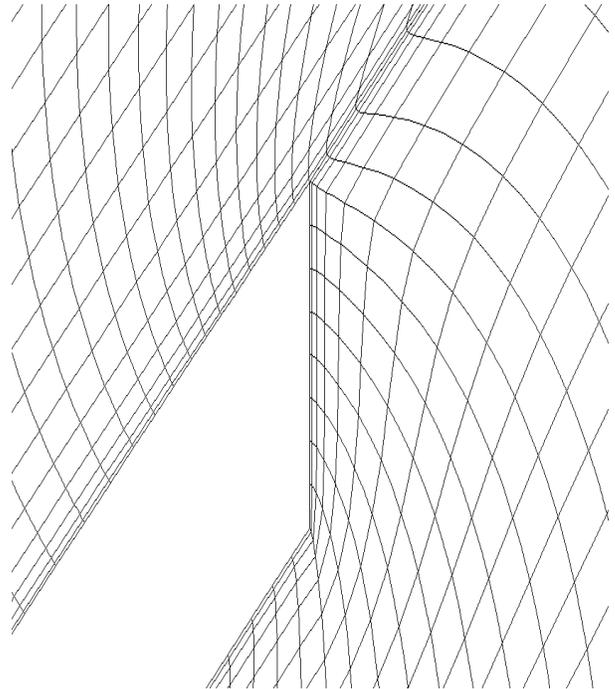


Figure 6b — Centrifugal impeller trailing edge.  
 $ibase = 1, nbase = 8, ibevel = 0.$

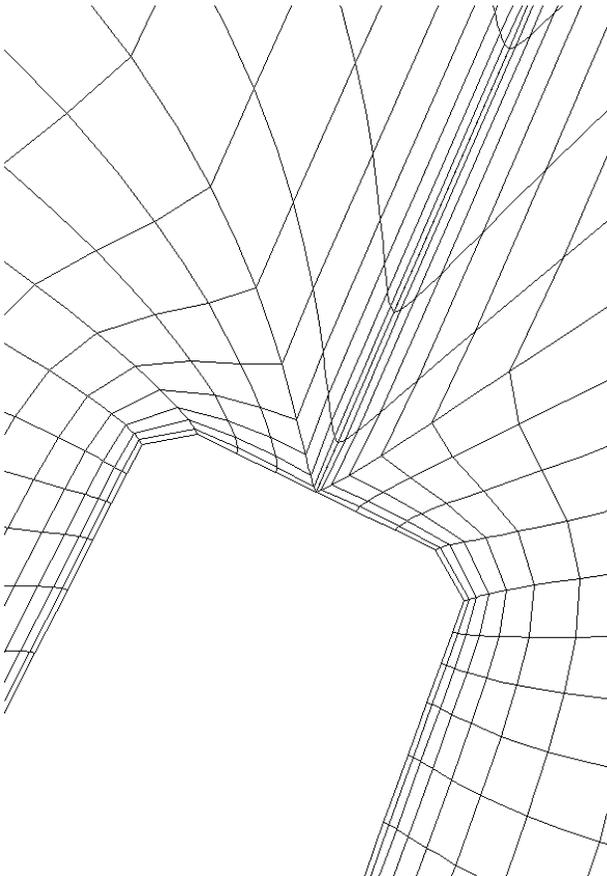


Figure 6c — Inlet guide vane trailing edge.  
 $ibase = 0, nbase = 8, ibevel = 1.$

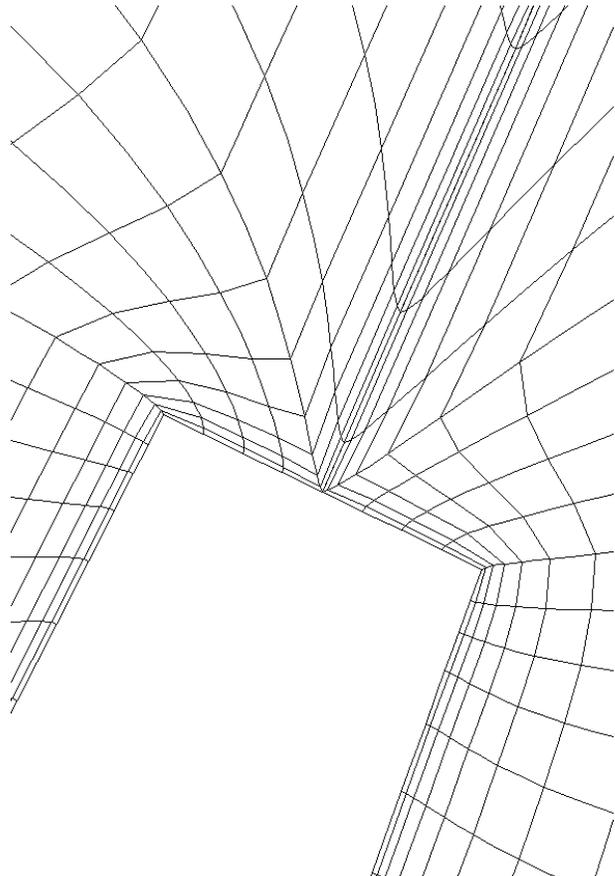


Figure 6d — Inlet guide vane trailing edge.  
 $ibase = 0, nbase = 8, ibevel = 0.$