

# GRAPE 2-D Grid Generator for Turbomachinery

## User's Manual and Documentation

Version 104, Oct. 24, 1997

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

The GRAPE code (GRids about Airfoils using Poisson's Equation) is an elliptic grid generator originally intended for isolated airfoils. The code was written by Reece Sorenson at NASA Ames Research Center (1, 2). This document describes modifications made to the GRAPE code to allow generation of periodic C-type grids for turbomachinery blades, and serves as the users' manual for turbomachinery problems. Reference (1) describes the theory behind the original GRAPE code, and reference (2) is a detailed users' manual for external flow problems. Any publications resulting from the use of this code should include these two references. Turbomachinery grids generated with the GRAPE code can be used directly with the RVCQ3D code. References (3, 4) describe the RVCQ3D code and include many examples of grids generated with the GRAPE code.

The GRAPE code allows arbitrary specification of inner and outer boundary points, then generates interior points as a solution to a Poisson equation. Forcing terms in the Poisson equation are chosen such that the desired grid spacing and intersection angles are maintained at the inner and outer boundaries.

The GRAPE code is based on a Cartesian  $(x, y)$  coordinate system which is mapped to a general body-fitted  $(\xi, \eta)$  coordinate system, where  $\xi$  is in the streamwise direction and  $\eta$  is in the blade-to-blade direction. Turbomachinery blades are often specified on a surface of revolution in a cylindrical-like  $(m, \theta)$  system, where  $m$  is the arc length along the surface and  $\theta$  is the circumferential direction. To use the GRAPE code for turbomachinery blades, all  $\theta$ -coordinates must be multiplied by some mean radius  $\bar{r}$  to give both coordinates consistent units of length. This can be done to the blade input coordinates in advance, or done within the GRAPE code using the variable *yscl*, described later. GRAPE can then use  $m$  as  $x$ , and  $\bar{r}\theta$  as  $y$ .

Numerous modifications were made to the original GRAPE code, although all input options described in (2) were retained. The code was rewritten to modernize the Fortran. A large number of *IF* and computed *GO TO* statements were eliminated, and *INCLUDE* and *PARAMETER* statements were added to simplify redimensioning. A few routines were speeded up for better performance on modern computers.

New inner and outer boundary routines were added for turbomachinery blades. The new inner boundary routine adds several input parameters that give considerable control over the spacing of points on the blade surface. Periodic outer boundaries are generated by shifting the mean camber line one-half pitch above and below the blades. Polynomial extensions are added upstream, and linear extensions are added downstream. The outer boundary points are equally spaced with arc-length and are periodic top-to-bottom.

GRAPE 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, and blade shapes input as  $(x, y)$  pairs. 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 describes how to dimension, compile, and run the GRAPE code for Silicon Graphics (SGI) workstations and Cray mainframes. The output file format is described. Finally all namelist input variables and blade coordinate input options are described in detail.

## References

1. 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.
2. 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.
3. Chima, Rodrick V., "Explicit Multigrid Algorithm for Quasi-Three-Dimensional Viscous Flows in Turbomachinery," *AIAA Journal of Propulsion and Power*, Vol. 3, No. 5, Sept.-Oct. 1987, pp. 397-405.
4. Chima, Rodrick V., "A  $k-\omega$  Turbulence Model for Quasi-Three-Dimensional Turbomachinery Flows," AIAA Paper 96-0248, Jan. 1995.

# Compiling and Running GRAPE

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

```
#!/bin/csh -f
cat > grape.f << `'/eof'
#GRAPE source code goes here
`'/eof'
cat > gridp << /eof
#gridp is an include file used for dimensioning the code
parameter(idm=385,jdm=64)
/eof
#compiler commands with options go here
/bin/rm...
```

On a unix platform, edit the script and go to the bottom. Make sure the dimensioning parameters statements are sufficient for the grid to be run, i.e., for a grid size of  $(jmax, kmax)$ , set  $idm \geq jmax$  and  $jdm \geq kmax$ . Comment, uncomment, or add compilation commands appropriate for the computer to be used. (See "Compiling GRAPE" below.) 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. Consult your compiler manual for compiling instructions.

## Compiling GRAPE

```
#SGI power series processor
f77 -pfa -O2 -lfpe -o grape grape.f
strip grape

#SGI R4000 processor
f77 -O2 -sopt -mips2 -lfpe -o grape grape.f
strip grape

#SGI R10000 compiler
f77 -O3 -mips4 -WK,-o=0,-so=2,-ro=0 -OPT:round=3:IEEE_arith=3 -lfastm
strip grape

#Cray C90
f90 -O3 -o grape grape.f
```

## Running GRAPE

The executable program is run as a standard unix process:

```
grape < std_input > std_output &
```

## Output Grid File

The output grid file is written to Fortran unit 1 (fort.1). The file is an unformatted binary file which may be linked to a file name before running grape,

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

or renamed after running grape,

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

Binary grid files can be used immediately by RVCQ3D 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 to convert the files to SGI binary. Files converted using *itrans* can be read into PLOT3D using the *read /binary* option <default>.

```
itrans 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
```

### Grid 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) jmax, kmax
      read(1)
      &( (x(j,k), j=1, jmax), k=1, kmax), ((y(j,k), j=1, jmax), k=1, kmax)
```

# Namelist Input

Only C-type grids can be generated for turbomachinery problems. Figures 1 – 3 at the end of this document show example grids distributed as test cases with the GRAPE code. Every other line in the  $j$ -direction has been omitted for clarity.

This section describes only those input parameters that are commonly used for turbomachinery problems. The GRAPE code does support many other options for generating inner and outer boundaries for isolated airfoils, and for specifying input parameters as arrays rather than constants. The user is referred to the original GRAPE documentation (2) for information on these other options.

GRAPE input consists of an ascii file with three blocks of namelist input. The first two blocks include parameters controlling grid size, inner and outer boundary stretching, and operation of the elliptic smoother. The third block contains blade coordinates entered as  $(x, y)$  pairs. Many of the input parameters are illustrated in figures 4 or 5.

Default values are set in a block data routine. NOTE: Many of the defaults are Sorenson's original values chosen for isolated airfoils and MUST be reset for turbomachinery problems. In the input description below, default values are given in angle brackets, <Default=value> or <default.> If no default is given the value must be input.

## **grid1 - Grid Size and Outer Boundary Parameters**

<i>jmax</i>	Grid size in the $j$ - (streamwise) direction (see fig. 4.) Typically 128-256, <default=100.>
<i>kmax</i>	Grid size in the $k$ - (blade-to-blade) direction (see fig. 4.) Typically 25 (inviscid) to 45 (viscous), <default=49.>
<i>jtebot</i>	$j$ -index of the lower trailing-edge point on a C-grid (see fig. 4,) <default=15.>
<i>jtebot</i>	MUST = $jmax+1-jtebot$ for turbomachinery problems, (see fig. 4,) <default=86.> $j$ -index of upper trailing-edge point on a C-grid.
<i>ntetyp</i>	MUST = 3 for turbomachinery problems, <default=1.> (Controls type of grid, O or C, in the full code.)
<i>nairf</i>	MUST = 5 for turbomachinery problems, <default=2.> (Controls inner boundary spacing in the full code.)
<i>nobshp</i>	MUST = 7 for turbomachinery problems, <default=1.> (Controls type of outer boundary in the full code.)
<i>jairf</i>	Number of blade input points entered in variables <i>airfx</i> and <i>airfy</i> in <i>grid3</i> (see fig. 5,) <default=0.>
<i>nibd</i>	Flag for type of clustering along the blade surfaces, span, and wake, <default=1.> = 6 Hyperbolic tangent clustering - smoothest, but may be sparse at blade center if <i>jmax</i> is small <default.> = 7 Hermite polynomial clustering - more uniform, but may grow too quickly near leading and trailing edges. Good for large <i>jmax</i> .
<i>dsi</i>	Grid spacing away from the inner boundary, in same units as blade input (see fig. 4) <default=0.01.> For inviscid solutions use $dsi \approx dsle$ or $dste$ , i.e., choose <i>dsi</i> to give square cells around the leading or trailing edges. For viscous solutions, use $dsi \approx chord/(10,000 \text{ to } 50,000)$ . Choose larger values for quick solutions, smaller values for accurate loss, skin friction, or heat transfer.
<i>xleft</i>	$x$ -coordinate of the inlet boundary of the grid (see fig. 4,) <default=-6.> Typically $xleft = \pm(0.5 \text{ to } 1.0) \times pitch$ . Smaller values give a more uniform grid, larger values give a stretched grid.
<i>xright</i>	$x$ -coordinate of the exit boundary of the grid (see fig. 4,) <default=6.> C-type grids can extend indefinitely downstream as long as <i>jtebot</i> and <i>jtebot</i> are large enough to cover the region downstream of the trailing edge.

<i>xle</i>	<i>x</i> -coordinate of the blade leading edge, (see fig. 4,) <default=0.> The input airfoil coordinates are rescaled to go from <i>xle</i> to <i>xte</i> . To prevent rescaling, set <i>xle</i> to the minimum value of the input blade coordinates.
<i>xte</i>	<i>x</i> -coordinate of the blade trailing edge, (see fig. 4,) <default=1.> The input airfoil coordinates are rescaled to go from <i>xle</i> to <i>xte</i> . To prevent rescaling, set <i>xte</i> to the maximum value of the input blade coordinates.
<i>rcorn</i>	Radius for the front corner of the C-grid (see fig. 4,) <default=1.> Typically $rcorn \approx pitch/8$ . Use $rcorn = 0$ . to give square corners.
<i>maxita</i> (2)	Maximum number of iterations on each grid level, <default=200,100> GRAPE supports multigrid convergence acceleration, but it is so fast on modern computers that multigrid is rarely used. <i>maxita</i> is an array with two elements. The first element refers to coarse grids and the second element refers to the finest grid. Generally <i>maxita</i> (1)=0 to skip the coarse grids and <i>maxita</i> (2)=100 to 300. Use <i>maxita</i> =0,0 to check initial grid spacings, boundary locations, etc.
<i>norda</i> (2)	Number of orders of magnitude to reduce the residuals for convergence, <default=4,1> <i>norda</i> is an array with two elements. The first element refers to coarse grids and the second element refers to the finest grid. Generally <i>norda</i> (1)=0 to skip the coarse grids and <i>norda</i> (2)=3 for three orders of magnitude reduction in the residual. GRAPE will usually terminate after <i>maxita</i> iterations before the <i>norda</i> limit is reached.
<i>nout</i>	MUST = 4 for turbomachinery problems, <default=1> Parameter controlling output options. <i>nout</i> =4 writes a binary grid to unit 1 in PLOT3D format.

### **grid2 - Grid Spacing and Algorithm Parameters**

<i>nobcas</i>	Controls angle between $\eta$ -grid lines and periodic (outer) boundary. This is an undocumented feature of the original GRAPE code. = 0 $\eta$ -lines are vertical at the periodic boundary <default, recommended> = 1 $\eta$ -lines are normal to the periodic boundary.
<i>pitch</i>	Cascade pitch in same units as blade input, (see fig. 4,) <default=1.>
<i>yscl</i>	Scale factor for blade input <i>y</i> -coordinates <default=1.> For an annular cascade the <i>y</i> -coordinates are input as $\bar{r}\theta$ . If $\theta$ -coordinates are input for <i>yairf</i> , use $yscl = \bar{r}$ to rescale the input. Note that <i>x</i> -coordinates can be rescaled using <i>xle</i> and <i>xte</i> .
<i>dsobi</i>	Grid spacing away from the periodic boundary, in same units as blade input (see fig. 4,) <default=0.2.> Use $dsobi \approx 0.5pitch/kmax$ to get nearly equal spacing at the periodic boundary.
<i>jcap</i>	Number of <i>j</i> -points on the inlet part of the C-grid, (see fig. 4.) Remaining points are distributed over the periodic boundaries. Increase <i>jcap</i> to pull points towards inlet, and vice-versa.
<i>nle</i>	Number of points equally-spaced around the blade leading edge, <default=15.>
<i>nte</i>	Number of points equally-spaced around the blade trailing edge, <default=10.> Should be an even number.
<i>dsle</i>	Spacing around the leading edge as a fraction of total arc length around blade <default=.0025.>
<i>dste</i>	Spacing around the trailing edge as a fraction of total arc length around blade, <default=.0025.>
<i>xtfrac</i>	Parameter that controls <i>x</i> -spacing away from the trailing edge, <default=1.>

The  $x$ -spacing away from the trailing edge is roughly  $xtfrac \times dste$ . Start with  $xtfrac=1$  and adjust if necessary.

- dswex*  $x$ -spacing at the grid exit, (see fig. 4,) <default is equally-spaced in  $x$ .>  
Only used if  $fwakex=1$  to stretch grid the downstream.  
The grid spacing along the wake cut stretches from  $xtfrac \times dste$  at the trailing edge to  $dswex$  at the exit.  $dswex$  is hard to estimate in advance, but should be roughly  $(xright - xte)/jtebot$ .
- dsra* (Pressure surface arc length)/(total surface arc length), <default=0.5.>  
Used to locate the center of the leading edge clustering on the blade. The clustering is centered about  $dsra \times$  (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.
- joble* and *jobte*  
The periodic outer boundary for a C-grid is made up of three segments, a polynomial segment upstream, the mean-camber line between the blades, and a linear segment downstream. Within GRAPE each segment is represented by an array of 10 or 11 points which are later reclustered. Variables *joble* and *jobte* are indices where the upstream and downstream segments connect to the mean-camber line, and can be used to manipulate the shape of the outer boundary to a limited extent. (See fig. 4)
- joble*  
Index on the mean-camber line where the upstream quadratic segment starts. Values can be <11>, 10, 9... The default <11> starts the upstream quadratic segment at the leading edge. Smaller values move the starting point inside the passage, which can be useful if the mean-camber line segment is distorted near the leading edge.
- jobte*  
Outer boundary index that connects to the trailing edge of the blade. Values can be ... 9, <10>, 11 ... The default <10> connects the trailing edge to points immediately above and below. Larger values move the connection inside the passage. Smaller values move the connection downstream, which may be useful if grid lines cross the trailing edge circles.
- fwakex*  
Flag for stretching the outer boundary grid spacing along the wake ( $j$ -direction).  
= 0 Equally-spaced outer boundary along the wake.  
= 1 Stretched outer boundary along the wake. Spacing at the trailing edge is set by  $xtfrac \times dste$ , spacing at the exit is set by  $dswex$ . <default.>
- kwakex*  
Flag for expanding the grid spacing across the wake ( $k$ -direction).  
= 0 Equally-spaced grid across the wake, <default.>  
= 1 Grid expands across the wake moving downstream. Probably gives better wake resolution downstream.
- aaai*, *bbbi*  
Exponents controlling the distance that angles and spacings at the inner boundary propagate into the interior. Small *aaai* and *bbbi* give large distances but slow convergence, and vice versa. Any value  $> 0$ . is acceptable. See [2] for details. <Default = 0.45>
- ccci*, *dddi*  
Like *aaai* and *bbbi*, but for the outer boundary, <default = 0.45.>
- csmo*  
Smoothing coefficient for periodic boundary <default=0.>  
Normally the outer (periodic) boundary points are fixed. If the resulting grid is too sheared at the outer boundary, setting  $csmo = (0.1$  to  $1.0)$  will smooth the outer boundary points with a Laplace-type filter after each iteration of the elliptic solver. This may improve the grid while sacrificing control of angles at the outer boundary. Non-zero values of  $csmo$  may prevent the GRAPE code from converging, although the grids may still be useful.

### ***grid3* – Blade Coordinates**

*airfx*      Array of  $x$ -coordinates around the blade, starting at the trailing edge, going clockwise around the blade, and repeating the first point (see fig. 5.) The coordinates are fit with a cubic spline and so must adequately define the leading and trailing edges.

*airfy*      Array of  $y$ - or  $\bar{r}\theta$  coordinates around the blade, ordered like *airfx*, (see fig. 5.)

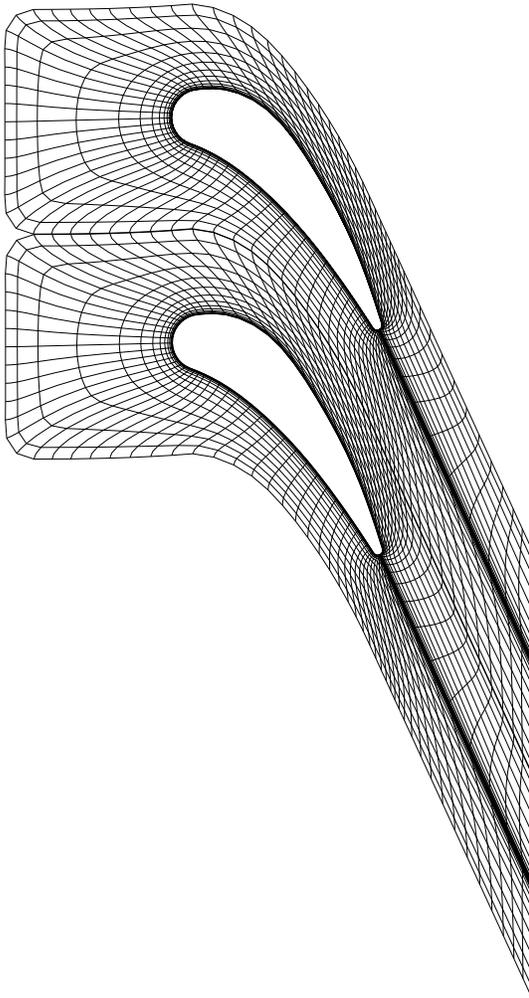


Figure 1 – 97 x 33 grid for a turbine vane

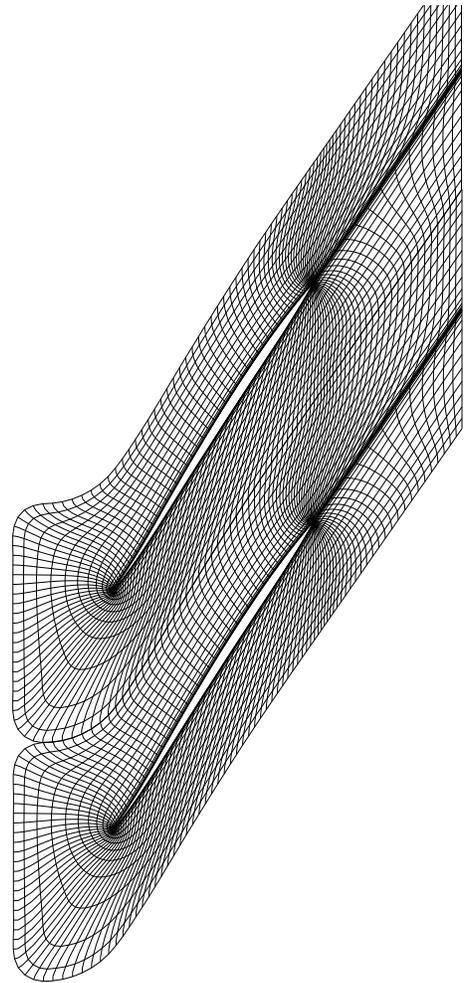


Figure 2 – 169 x 33 grid for a compressor rotor

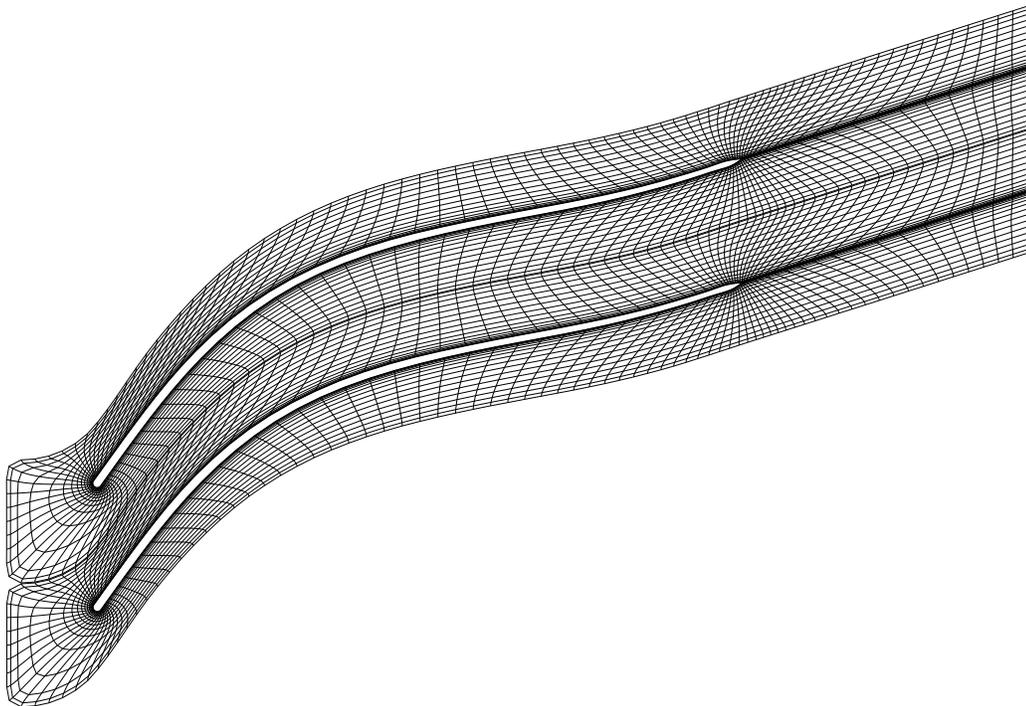


Figure 3 – 161 x 33 grid for a centrifugal impeller rotor

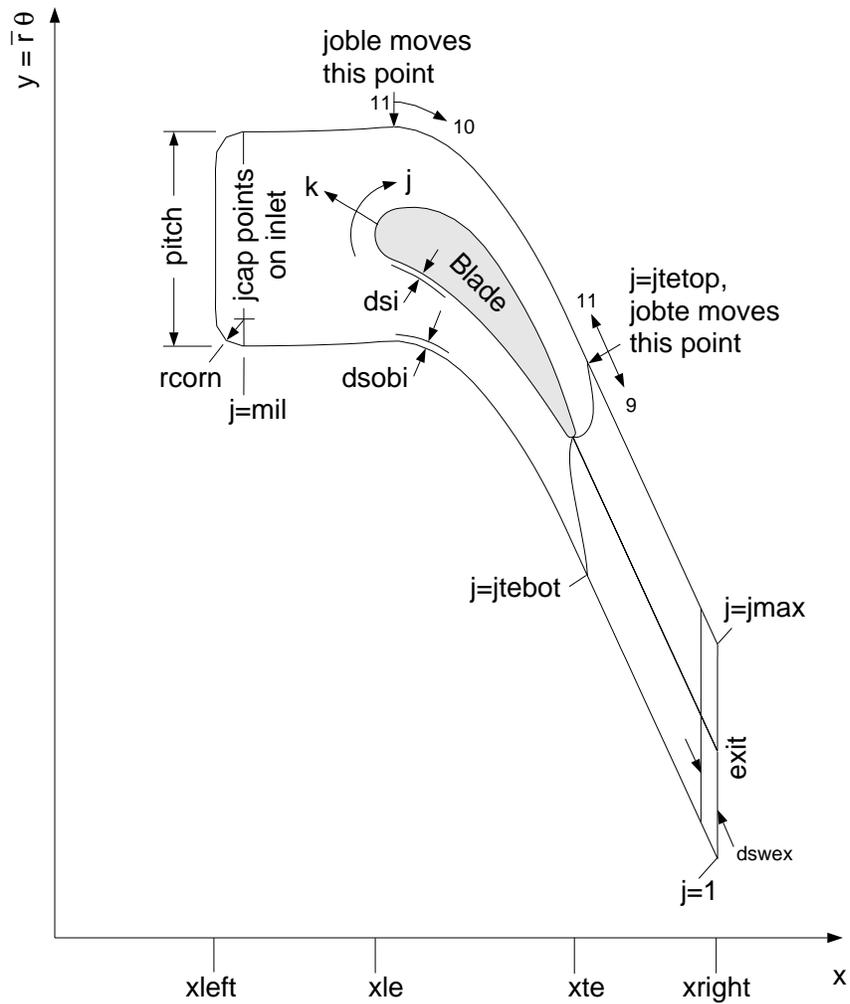


Figure 4 – GRAPE nomenclature and input variables

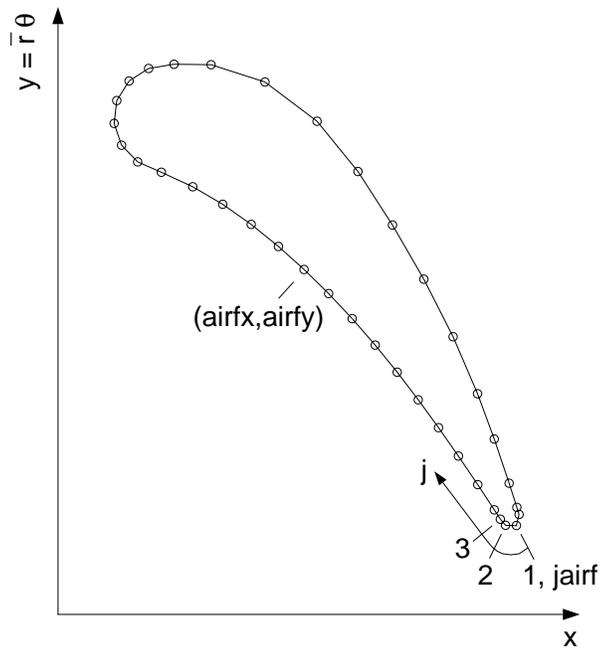


Figure 5 – Blade coordinate input variables