

# SWIFT - Multiblock Analysis Code for Turbomachinery User's Manual and Documentation

Version 400, August, 2011

Dr. Rodrick V. Chima  
NASA Glenn Research Center, MS 5-12  
21000 Brookpark Road  
Cleveland, Ohio 44135 USA

phone: 216-433-5919

fax: 216-433-5802

email: Chima@nasa.gov

internet: <http://www.grc.nasa.gov/WWW/5810/rvc>

download: <http://sr.grc.nasa.gov>

## Introduction

SWIFT is a multiblock computational fluid dynamics (CFD) code for analysis of three-dimensional viscous flows in turbomachinery. It solves the thin-layer Navier-Stokes equations using explicit finite-difference techniques. It can be used to analyze linear cascades or annular blade rows with or without rotation. Three differencing schemes are available – a central difference scheme with artificial viscosity [1, 2], and the AUSM+ [3, 4] and H-CUSCP [4, 5, 6] upwind schemes. Three turbulence models are available – the Baldwin-Lomax [7, 8] and Cebeci-Smith [8, 9] algebraic models, and Wilcox's 2006  $k-\omega$  model with a stress limiter [10, 11].

The code uses an explicit multistage Runge-Kutta solution scheme to march the solution in time from an initial guess to a steady-state solution [12]. A spatially varying time step and implicit residual smoothing are used to accelerate convergence. Preconditioning can be used to accelerate convergence for low speed flows [13, 14].

Limited multi-block capability can be used to model complicated geometries. C-type grids are used to give good resolution of blade leading edges and wakes. H-grids can be used to extend the domain upstream, and O-grids can be used to resolve hub and tip clearance regions. A mixing plane technique that uses characteristic boundary conditions can be used for multistage machines [15]. H-grids can also be used to analyze isolated blade rows or internal duct flows.

Grid input is in standard PLOT3D xyz-file format. Grids are usually generated using TCGRID [16], a turbomachinery grid code developed by the author. TCGRID is distributed by the NASA Glenn Research Center Software Repository <http://sr.grc.nasa.gov>.

SWIFT is written completely in Fortran and runs quickly on a Linux workstation. SWIFT will run on a PC or Mac, but the user will have to make the appropriate conversions. Parallel processing using OpenMP directives gives excellent performance on multi-core shared memory computers. Solution files are compatible with most CFD flow visualization packages.

Six test cases are included with SWIFT and TCGRID:

- Goldman's annular turbine vane [1, 4, 17]
- The space shuttle main engine (SSME) two-stage fuel turbine [15, 18, 19]
- The NASA large, low-speed centrifugal impeller [14, 20]
- Core compressor rotor 37 [21, 22, 23, 24]
- Core compressor stage 35 [21, 22, 25]
- Transonic fan rotor 67 [2, 25]

Grids for the turbomachinery test cases must be generated with the TCGRID code that is distributed separately. Comparisons with experimental data are included in separate Excel files.

This report serves as the user's manual and documentation for SWIFT. The code and some aspects of the numerical method are described. Steps for code installation and execution are given for Linux systems. The grid, input, and output variables are described in detail.

## **Features of SWIFT**

### **Applications**

Linear cascades  
Axial compressors and turbines  
Centrifugal impellers and mixed-flow machines (but no splitters)  
Radial diffusers  
Pumps  
Rectangular ducts  
Isolated blade rows or multistage machines  
Hub and tip clearances

### **Multi-block Capability**

C-grids around blades  
H-grid upstream  
O-grids in hub- or tip-clearance regions (or periodic clearance model)  
Mixing-planes between blade rows  
Discontinuous grids at mixing planes added in version 400  
H-grids for blades or ducts

### **Formulation**

Navier-Stokes equations written in Cartesian coordinates with rotation about the x-axis  
Thin-layer equations in streamwise direction, all cross-channel viscous terms retained  
Central-difference, AUSM<sup>+</sup>, and H-CUSP differencing schemes for inviscid terms  
Central-difference scheme for viscous terms

### **Turbulence Models**

Baldwin-Lomax (algebraic)  
Cebeci-Smith (algebraic)  
Wilcox's 2006 two-equation k- $\omega$  model with stress limiter and cross-diffusion terms  
Transition and surface roughness effects in all models

### **Numerical Method**

Explicit multi-stage Runge-Kutta scheme  
Variable time-step and implicit residual smoothing for convergence acceleration  
Preconditioning for low-speed (incompressible) flows

### **Input**

General grid files in PLOT3D format, usually generated using TCGRID  
Namelist input of flow parameters

### **Printed Output**

Residual history  
Spanwise output of circumferentially averaged flow quantities at the grid inlet and exit  
Streamwise output of blade row performance and blade surface properties  
Printed output can be edited manually and plotted with Microsoft Excel or other line plot software

### **Computer Requirements**

Fortran 90 compatible compiler  
Runs as a quick batch process on Linux, Mac, or Windows computers  
Parallel processing on multi-core, shared memory computers using OpenMP directives  
Solution times range from one to several hours on multi-core computers  
Dynamic memory allocation reduces memory requirements and avoids recompiling for most problems

### **Graphical Output**

No graphical output is provided with SWIFT, but a separate CFD visualization package is needed to view and evaluate the solutions. Grid, solution, and  $k-\omega$  files are written in standard PLOT3D format and can be read directly and plotted with the public-domain CFD visualization tool PLOT3D or the commercial tools FieldView, TecPlot, or EnSight CFD. Check the following web sites for more information.

PLOT3D : <http://www.nas.nasa.gov/Research/Software/swdescription.html>

TecPlot: <http://www.tecplot.com/>

FieldView: <http://www.ilight.com/>

EnSight CFD: <http://www.ensightcf.com/>

## Numerical Method

### Multistage Runge-Kutta Scheme

Jameson, Schmidt, and Turkel developed multistage schemes [12] 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.

The  $k$ th-stage of an  $n$ -stage scheme may be written as:

$$q^k = q^0 - \alpha^k \Delta t (R_r^k + R_v^0)$$

where  $q$  is an array of five conservation variables (see *Solution Q-File*, pp. 28),  $k$  is current the stage,  $q^0$  is the previous time step,  $\alpha^k$  are the multistage coefficients discussed below,  $\Delta t$  is the time step,  $R_r^k$  is the inviscid part of the residual, and  $R_v^0$  is the viscous part of the residual plus the artificial dissipation, if applicable. Note that  $R_r^k$  is evaluated at every stage  $k$ , but  $R_v^0$  is only evaluated at the initial stage for computational efficiency.

n	$\alpha^1$	$\alpha^2$	$\alpha^3$	$\alpha^4$	$\alpha^5$	$\lambda^*$
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 and maximum Courant number for  $n$ -stage schemes.

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

The number of stages is set with the variable *nstg*. *Nstg* = 4 is recommended, although Jameson et al. tend to favor 5 stages. The 2-stage scheme (*nstg* = 2) is very robust and often works when everything else fails.

A spatially-varying  $\Delta t$  is used to accelerate the convergence of the code. Setting *ivdt* = 1 sets the Courant number to a constant (input variable *cf1*) everywhere on the grid, and recalculates it every *icrnt* iterations. This option is strongly 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, which can cause jumps in the residual if *icrnt* is too big.

Implicit residual smoothing (described later) can 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 central-difference scheme is selected by setting *icdup* = 1 (the default.) It is the fastest of the three differencing schemes used in SWIFT, it gives moderately smeared shocks, and it may show velocity overshoots at the edge of boundary layers. It is recommended when quick answers are desired.

Second-order central differences require 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 2.0. A good starting value is 1.0. If the solution is wiggly, increase *avisc4* by 0.25. If it is smooth, try reducing *avisc4* by 0.25. 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, et al. [12] are used to reduce overshoots around shocks. The input variable *avisc2* scales the second difference artificial viscosity. *Avisc2* can be set to 0.0 for purely subsonic flows, and is usually set to 1.0 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 a few cells regardless of the value of *avisc2*. The magnitude of *avisc2* 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 the 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.

For computational efficiency the artificial viscosity is usually calculated only at the first stage of the Runge-Kutta scheme. This works well for most problems, but for difficult problems the robustness of the scheme can be improved by updating the artificial viscosity more often. This is selected by setting *ndis* = 2. For *nstg* = 2 or 3 this has no effect. For *nstg* = 4 the artificial viscosity is calculated at stages 1 and 2. For *nstg* = 5 the artificial viscosity is calculated at stages 1, 3, and 5. The physical viscous terms are calculated at the same time as the artificial viscosity. Thus, setting *ndis* = 2 increases the CPU time per stage significantly, but it is so reliable that it is usually the preferred scheme.

### **AUSM<sup>+</sup> Upwind Scheme**

The Advection Upstream Splitting Method (AUSM<sup>+</sup>) upwind scheme is selected by setting *icdup* = 1. It is the least dissipative but slowest of the three differencing schemes used in SWIFT. It is recommended for most problems.

The AUSM<sup>+</sup> family of upwind schemes was developed by Meng-Sing Liou and others [3, 4]. The AUSM<sup>+</sup> scheme defines a cell interface Mach number based on characteristic speeds from neighboring cells. The interface Mach number is used to determine the upwind extrapolation for the convective part of the inviscid fluxes. A separate splitting is used for the pressure terms. The van Albada limiter is used to estimate interface fluxes with second-order accuracy.

The speed of sound at the cell interface is multiplied by a function that effectively scales the numerical dissipation with the local flow speed, giving appropriate amounts of dissipation for all flow speeds. The scaling function is based on an average interface Mach number that must be limited by a cutoff relative Mach number  $M'_{ref}$ . Reference Mach numbers for rotors and stators are set using input variables *refmr* and *refms*, which should be the largest relative Mach number expected in those blade rows, but limited to a maximum of 1.0. The reference values are then reduced by the factor *ausmk* = 0.2 – 1.0.

### **H-CUSP Upwind Scheme**

The Convective Upwind Split Pressure (CUSP) upwind scheme is selected by setting *icdup* = 2. It gives the sharpest shocks but is the most dissipative of three differencing schemes used in SWIFT. It is second in execution speed. It is only recommended for occasional problems where the AUSM<sup>+</sup> scheme will not converge.

CUSP schemes were described by Tatsumi, Martinelli, and Jameson in [5, 6]. The H-CUSP scheme uses the stagnation enthalpy *h* as the conservation variable in the energy equation. The scheme was developed as a flux-split scheme similar to AUSM<sup>+</sup>, but it is implemented as a limited dissipative flux added to a central-difference scheme. Jameson's SLIP limiter is used to produce a second-order non-oscillatory scheme. For computational efficiency the dissipative fluxes are updated less often than the central-difference fluxes. The implementation in SWIFT is described in [4].

Like the AUSM<sup>+</sup> scheme, the H-CUSP scheme requires a cutoff Mach number  $M'_{ref}$ . Reference Mach numbers for rotors and stators are set using input variables *refmr* and *refms*, which should be the largest relative Mach number expected in those blade rows, but limited to a maximum of 1.0. The reference values are then reduced by the factor *hcuspk* = 0.05 – 0.20.

### **Implicit Residual Smoothing**

Implicit residual smoothing is selected by setting *irs* = 1. It 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 to 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:

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

Here  $\varepsilon_\xi$ ,  $\varepsilon_\eta$ , and  $\varepsilon_\zeta$  are constant smoothing coefficients in the body-fitted coordinate directions  $\xi$ ,  $\eta$ , and  $\zeta$ , shown in Fig. 1,  $\delta_{\xi\xi}$  is a second-difference operator,  $\bar{R}$  is the smoothed residual, and  $R$  is the unsmoothed residual.

It can be shown that implicit residual smoothing does not change the solution if the scheme converges. Linear stability theory shows that the scheme can be made unconditionally stable if the  $\varepsilon_i$  are big enough, but also shows that the effects of artificial viscosity are diminished as the Courant number is increased. In practice the best strategy is 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  $\varepsilon_i$  needed for stability are given by:

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

where  $\lambda^*$  is the Courant limit of the unsmoothed scheme (given in Table 1), and  $\lambda$  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 SWIFT. The 1-D limit for  $\varepsilon$  given above usually works well, but can be increased up to  $2.5 \times$  if the solution blows up, and can be decreased slightly to improve convergence if the solution is stable. Values of  $\varepsilon$  are scaled within the code at each grid point by multiplying  $eps$  by the same Eigenvalue scaling coefficients used for the artificial dissipation. This has proven to be quite robust.  $epi$ ,  $epj$ , and  $epk$  can be used to scale  $eps$  in the i-, j-, and k-directions, but they are usually left at their default values of 1.0.

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) by setting  $irs = 2$ , but  $eps$  must be increased (approximately doubled.) This option is rarely used.

### **Preconditioning**

Density-based schemes like SWIFT solve the continuity equation by driving the density residual to zero. For low speed (nearly incompressible) flows the density residual is physically near zero, and so the schemes fail to converge. Preconditioning, described by Turkel in [13], 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,  $W = [p, u, v, w, h]$ , 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. Preconditioning works extremely well for the Euler equations and less well for the Navier-Stokes equations. It allows solutions to be run at very low flow speeds that simply would not converge otherwise.

It can be difficult to get a solution started with preconditioning. It is usually best to run 200-500 iterations with preconditioning turned off, then run to convergence with preconditioning turned on.

The preconditioning operator is designed so that it has no effect on the steady-state solution; however, for the central-difference scheme preconditioning also modifies the artificial dissipation operator. This tends to reduce the artificial dissipation when preconditioning is used.

The preconditioning matrix has the local relative Mach number in the denominator and must be by a cutoff Mach number  $M'_{ref}$ . Reference Mach numbers for rotors and stators are set using input variables  $refmr$  and  $refms$ , which should be the largest relative Mach number expected in those blade rows, but limited to a maximum of 1.0. The reference values are then reduced by the factor  $pck = 0.1 - 0.3$ . The solution will diverge quickly if these parameters are too small, and may converge slowly if they are too big.

At low Mach numbers the default inlet boundary condition may not work properly. If contour plots of pressure or velocity look bad near the inlet, set  $ibcinu=2$  to extrapolate meridional velocity at the inlet.

## Recommended Numerical Parameters

nstg	ndis	cfl	eps	Comments
2	1	2.5	1.35–1.60	very robust, fast per stage
4	2	5.6	0.75–1.00	good overall scheme
5	2	7.0	1.25–1.50	slow per stage, fast convergence

Table 2 Recommended numerical parameters for three Runge–Kutta schemes.

Table 2 lists recommended parameters for three numerical schemes, in order of the author’s preference. For all schemes use  $irs = 1$ . For the central-difference scheme use  $avisc2 = 1.0$  and  $avisc4 = 0.5$ .

## Turbulence Models

The turbulence model is selected using input variable  $ilt$  (Inviscid, Laminar, Turbulent,) see *&nam5 – Viscous Parameters*, pp. 20.

Three turbulence models are available in SWIFT – the Baldwin-Lomax model, the Cebeci-Smith model, and Wilcox’s 2006 two-equation  $k-\omega$  model. All three models include transition models and surface roughness effects.

### Baldwin-Lomax and Cebeci-Smith Turbulence Models

The Baldwin-Lomax model is selected by setting input variable  $ilt = 2$ . The model is implemented as described in the original reference [7], except that two constants have been changed to  $C_{cp} = 1.216$  and  $C_{Kleb} = 0.646$ . The length scale for the Baldwin-Lomax model is correlated to the maximum of a function  $f = y|\omega|D$ , where  $y$  is the distance from the wall,  $|\omega|$  is the magnitude of the vorticity, and  $D$  is the Van Driest damping function. In some cases that maximum is not well behaved, so SWIFT limits the search to grid lines at indices  $jedge$  away from the blade,  $kedgh$  from the hub, and  $kedgt$  from the tip. Indices  $jedge$ ,  $kedgh$ , and  $kedgt$  should be chosen slightly larger than the largest extent of the boundary layer. Solutions are usually insensitive to values of these parameters if they are big enough, but may under predict viscous effects if they are too small.

The Cebeci-Smith model is selected by setting input variable  $ilt = 3$ . The model is implemented like the Baldwin-Lomax model except that the length scale is found by integrating

$$\int_0^{\delta} f dy = \delta^* u_e$$

as described in [8]. Here the upper bound of the integral is usually found automatically since  $f \rightarrow 0$  as  $y \rightarrow \delta$ . However, cases with free-stream vorticity can have a non-zero  $f$  outside the boundary layer, so again input variables  $jedge$ ,  $kedgh$ , and  $kedgt$  are used to bound the integrals in SWIFT. The Cebeci-Smith model is very reliable for turbine heat transfer problems, but is not recommended for transonic compressors that may produce free-stream vorticity behind the bow shock.

Transition is predicted in both models at the location where  $\mu_{turb}/\mu_{lam} > cmutm$ , where  $cmutm$  is an input variable usually set to 14, as recommended in [7]. The model is crude but often works surprisingly well for moderate Reynolds numbers and low free-stream turbulence.

Roughness effects are included in both models using the Cebeci-Chang roughness model [9]. The model modifies the turbulent length scale based on the equivalent sand-grain roughness height in turbulent wall units  $h^+$ . The roughness height is input using variable  $hrough$ , and  $h^+$  is calculated internally. If  $hrough = 0.0$  a model for a hydraulically smooth surface is used.

### Wilcox $k-\omega$ Turbulence Model

The  $k-\omega$  model is selected by setting input variable  $ilt = 4$  or  $5$ . The model is described in [10], and implemented as described in [11] using a first-order upwind ADI scheme. Two versions of the model are included, a baseline model ( $ilt = 4$ ) and a low Reynolds number model ( $ilt = 5$ ). The baseline model gives a fully

turbulent solution that is valid all the way to the wall, unlike k- $\epsilon$  models. The low Reynolds number model includes transition effects.

Wilcox's 2006 model includes a cross-diffusion term that reduces dependence on freestream values of  $\omega$ , and a shear stress limiter that reduces the turbulent viscosity when production of turbulent kinetic energy exceeds destruction. The shear stress limiter has been shown to improve results for shock-separated flows. The stress limiter is selected by setting  $isst = 1$ .

Three input parameters affect the k- $\omega$  model, the surface roughness  $hrough$  as described above, the free-stream turbulence level  $tintens$  (typically 0.0 to 0.05), and the free-stream turbulent viscosity  $tmuinf = (\mu_{turb}/\mu_{0r})_m$ , typically 0.1. Solutions are generally insensitive to  $tmuinf$ , except for the location of transition.

Previous versions of SWIFT used a turbulent length scale  $tlength$  instead of  $tmuinf$ .  $Tlength$  was awkward to use but has been retained in the code for backward compatibility.

## Nondimensionalization

Ref. State	English Units	SI Units
$P_{0r}$	2116.8 lb <sub>f</sub> /ft <sup>2</sup>	1.0135 x 10 <sup>5</sup> Pa
$T_{0r}$	519 R	288.3 K
$c_{0r}$	1116.7 ft/sec	340.39 m/sec
$\rho_{0r}$	.0765 lb <sub>m</sub> /ft <sup>3</sup>	1.2246 kg/m <sup>3</sup>
$\rho_{0r}c_{0r}$	85.427 lb <sub>m</sub> /sec/ft <sup>2</sup>	416.8416 kg/sec/m <sup>2</sup>
$\mu_{0r}$	1.197 x 10 <sup>-5</sup> lb <sub>m</sub> /(ft sec)	1.71 x 10 <sup>-5</sup> kg/(m sec)
$renr$	7.137 x 10 <sup>6</sup> [1/ft]	2.437 x 10 <sup>7</sup> [1/m]
	5.947 x 10 <sup>5</sup> [1/in]	2.182 x 10 <sup>5</sup> [1/cm]

Table 3. Standard reference quantities usually used for nondimensionalization.

The grid xyz-file may be input in arbitrary units of length. The input parameters to SWIFT and the variables in the output q-file are all nondimensional except for lengths, which have the same units as the grid.

All quantities are nondimensionalized by an arbitrary reference stagnation state defined by stagnation density  $\rho_{0r}$ , sonic velocity  $c_{0r} = \sqrt{\gamma RT_{0r}}$ , and laminar viscosity  $\mu_{0r}$ . Standard atmospheric conditions, given in Table 3 above, are often used for the reference state for a compressor in a test cell. However, any self-consistent state may be used as long as the units of length are consistent with the grid units. For example, for a fan in an engine at flight conditions it is useful use freestream total conditions for the reference state.

Input pressures and temperatures are nondimensionalized by  $P_{0r}$  and  $T_{0r}$ , respectively. Within the code pressures are usually nondimensionalized by  $\rho_{0r}c_{0r}^2 = \gamma P_{0r}$ . Inlet pressures and temperatures are nondimensionalized similarly, so that  $P_{0in} = T_{0in} = 1.0$  when the inlet is at standard conditions. However,  $P_{0r}$  and  $T_{0r}$  can also be set arbitrarily using the initial condition input (see *Initial Condition Input*, pp 24) or a qin file (see *Inlet and Exit Profiles*, pp. 29). Input velocities are sometimes nondimensionalized by  $c_{0r}$ , but are usually input as a Mach number.

The reference state defines a reference Reynolds number  $renr$  that must be input to SWIFT (see *&nam5 – Viscous Parameters*, pp. 20).  $renr = \rho_{0r}c_{0r} / \mu_{0r}$  has units of [1/grid units].  $renr$  is the same for all cases with the same reference state and grid units.

Output quantities should be self explanatory, except for the mass flow. The mass flow may be output with the residual history (see variable *mioe* under *&nam6 – Output Control*, pp. 22). Mass flow is also output in the tables labeled “theta-averaged quantities,” at the bottom of the column labeled “% mdot.” In either case, the mass flow is nondimensionalized by  $\rho_{0r}c_{0r}$  and has units of [grid units]<sup>2</sup>. The mass flow through the full annulus is given (rather than mass flow per passage,) so that the printed mass flow should be constant through a multistage machine.

## Calculations for liquids

### Nondimensionalization

Nondimensionalize using conditions for liquids, but calculate  $c_{0r}$  as if for air.

R	= 1716.58 ft <sup>2</sup> /(sec <sup>2</sup> R)	ideal gas constant
$\gamma$	= 1.4	$C_p / C_v$
$T_{0r}$	= 60 F = 519 R	
$c_{0r}$	= 1116.7 ft / sec	
$\rho_{0r}$	= 62.37 lb <sub>m</sub> / ft <sup>3</sup>	
$P_{0r}$	= $\rho_{0r} R T_{0r} = 1,725,644$ lb <sub>f</sub> / ft <sup>2</sup>	
$\nu_{0r}$	= 1.217e-5 ft <sup>2</sup> / sec	kinematic viscosity for water at 60 F
vispwr	= 1	laminar viscosity ~ T
renr	= $c_{0r} / \nu_{0r} = 7.646e6$ / in	(convert to appropriate grid units)
om	= omega [rad / sec] / $c_{0r}$	(convert to appropriate grid units)

### Initial Conditions

Calculate the inlet and exit velocities  $V_1$  and  $V_2$  from the flow rate  $Q$  and areas  $A$  using

$$V = Q / A .$$

Approximate the Mach numbers for the initial conditions using

$$M \approx V / c_{0r} .$$

Calculate the total pressure rise  $\Delta P_0$  from the head rise  $H$  using

$$\Delta P_0 = \rho g H .$$

Calculate the pressure ratios for the initial conditions using

$$P_{02} / P_{01} = (P_{01} + \Delta P_0) / P_{01} .$$

Calculate the temperature ratios for the initial conditions using

$$T_{02} / T_{01} = P_{02} / P_{01} .$$

Calculate the static pressure rise  $\Delta P$  using

$$\Delta P = \Delta P_0 - 0.5 \rho (V_2^2 - V_1^2)$$

Calculate  $prat$  using

$$prat = P_2 / P_{01} = (P_1 + \Delta P) / P_{01} .$$

This should give a solution close to the correct flow rate, but you will probably have to run several cases with different values of  $prat$  to get the flow rate exactly.

### Preconditioning

Run SWIFT 200 – 500 iterations with no preconditioning, using:

icdup=0, nstg=2, avisc2=1, avisc4=1, cfl=2.5, eps=1.5, ibcinu=1, ipc=0.

Then restart with preconditioning turned on (see *Preconditioning*, pp. 6), with:

icdup=0, nstg=2, avisc2=1, avisc4=1, cfl=2.5, eps=1.5, ibcinu=1, ipc=1, refmr=.15, pck=.30.

The AUSM<sup>+</sup> scheme works well at low speeds. Try:

icdup=1, nstg=2, cfl=2.5, eps=1.5, ibcinu=1, ipc=1, refmr=.15, pck=.30, ausmk=0.3.

## Grids

SWIFT can handle single-block grids and a limited variety of multi-block grids. Grids are usually generated using TCGRID [16]. Dummy grid lines are used to handle periodic boundary conditions and transfer of information between blocks, and must be included in the grid file. All grid types currently supported by SWIFT will have a dummy grid line at  $j=j_m$ , except for grids in rectangular ducts which have no dummy grid line. Grids are stored in standard PLOT3D format (see *Grid XYZ-File*, pp. 28).

The connectivity between the grids is specified using an index file (see *Index File*, pp. 25). In TCGRID, setting *iswift=1* in namelist 5 will automatically add dummy grid lines and produce a preliminary index file.

### C-grids (Blades)

The basic SWIFT grid consists of a C-type grid around a blade, as shown in Figure 1. The i-direction goes from  $i=1$  at the lower exit to  $i=i_m$  at the upper exit. The j-direction goes from  $j=1$  at the blade to  $j=j_m-1$  at the periodic boundary ( $j=j_m$  at the dummy grid line.) The k-direction goes from 1 at the hub to  $k=k_m$  at the tip. Calculations run with a single grid avoid some data I/O and thus run about 10 percent faster than a multiblock grid with the same number of points.

### H-grids (Upstream, Blades, Rectangular Ducts)

Three types of H-grids are supported in SWIFT v.300. The type of H-grid is flagged by index file variable *i1* (see *Index File*, pp. 25).

1. An H-grid can be added to extend a C-grid upstream, as shown in Figure 2. A dummy grid line is needed at  $j=j_m$  to apply the periodic boundary conditions. Flagged by setting  $i1 = 0$ .
2. An H-grid can also be used inside a rectangular duct (not shown.) No dummy grid lines are needed. Multiblock grids are not supported with this grid type. Flagged by setting  $i1 = 1$ .
3. A single H-grid can be used inside a blade passage, as shown in Figure 3. A dummy grid line is needed at  $j=j_m$  to apply the periodic boundary conditions. Multiblock grids are not supported with this grid type. Flagged by setting  $i1 = \text{leading edge index} > 1$ .

In each case the i-index goes from inlet to exit, the j-direction goes from blade to blade, and the k-direction goes from hub to tip.

### O-grids (Hub and Tip Clearance Gaps)

O-type grids can be used to resolve the hub or tip clearance regions of blade (visible in Figure 5). Clearance regions can also be modeled using a simple periodic boundary condition that does not require gridding the region. For O-grids the i-direction starts at the trailing edge cut and wraps around the O. The j-direction starts at the center line cut and goes to the perimeter of the O.  $j=j_m$  is a dummy grid line that overlaps the connecting C-grid by one point. The k-direction goes from the hub to the blade for hub clearances, or from the blade to the casing for tip clearances.

### Multistage Grids

Multistage C-grids are generated one blade row at a time using TCGRID. The individual grids must meet certain requirements:

1. Use identical hub and tip coordinates for all blade rows.
2. The blades must be in the correct location and orientation. Use *ztrans* to move the blades axially, and *tflip* to flip the  $\theta$ -coordinates if necessary.
3. The grids must match at an interface between the blades. Set the exit boundary coordinates of grid 1 to the inlet boundary coordinates of grid 2. Place the interface midway between the blades, or close enough to blade 2 to get a good C-grid.
4. For SWIFT, the grids must overlap exactly one cell at the interface. On grid 1 set *dswex* to give a fine spacing near the exit, and set *dslap* = *dswex*. This resets the grid spacing at the exit from approximately *dswex* to exactly *dslap*. On grid 2 set  $dsmax_2 = dslap_1$ . This will cause the dummy grid line from grid 2 to overlap grid 1 by *dsmax*.
5. The relative circumferential spacing between the grids does not matter.

SWIFT version 400 allows non-point-matched grids at mixing planes. Neighboring grids can have different numbers of points and discontinuous spacings in the spanwise direction. However, using continuous grids across mixing planes may improve solution continuity, and will simplify printed and graphical output at a given spanwise location.

Grids and index files from neighboring blade rows are merged into a multi-block PLOT3D file using a Fortran program called `multix.f`. The merged index file will have the correct block sizes and key indices, but must be edited manually to set block connectivity, and options for mixing planes, clearances, and endwall rotation. Additional details about generating multistage grids are given in the TCGRID user's manual [16].

## Unzipping, Compiling, and Running SWIFT

SWIFT is supplied as a zipped file. It will unzip into a directory with the same name as the file. This documentation should be in the main directory. There are subdirectories for source code and test cases. On a Linux system:

```
unzip swift_400.zip
```

### Compiling SWIFT

Go to the src directory and edit the Makefile. Compiler commands are set for the Intel Linux compiler,  
`FC = ifort -O3 -ipo -xP -parallel -openmp`

Change the commands as necessary for your compiler. Here

```
-O3          gives full optimization
-ipo        enables interprocedural optimization
-xP        optimizes for Intel Core architectures
-parallel   generates parallel code
-openmp    tells the compiler to use the OpenMP directives in SWIFT
```

Near the bottom of the Makefile there may be a line that moves the executable to a bin directory. Keep or remove this line as desired.

```
mv swift ~/bin
```

Save the file and exit.

Most arrays are allocated dynamically, but a few work arrays have fixed maximum dimensions. Edit `modules.f90` and modify the maximum dimensions in module `maxsize` if desired. Default values for most input variables are also set here.

```
integer, parameter::ni=255, nj=63, nk=63
```

Run `make`. Move the executable `swift` to a directory in your path.

Clean up object and executable files if desired by running

```
make clean
```

### File Names

Unit	Default name	Description	Reference
fort.1	grid.xyz	grid file from TCGRID	Grid XYZ-file, pp. 28
fort.2	q_in.q	binary input solution file, read if <i>iresti=1</i>	Solution Q-File, pp. 28
fort.3	q_out.q	binary output solution file, written if <i>iresto=1</i>	Solution Q-File, pp. 28
fort.10	index.dat	text index file, required	Index File, pp. 25
fort.7	kw_in.kw	binary input k- $\omega$ file, read if <i>ilt = 4 or 5</i>	Turbulence Model k- $\omega$ file, pp. 28
fort.8	kw_out.kw	binary output k- $\omega$ file, written if <i>ilt = 4 or 5</i>	Turbulence Model k- $\omega$ file, pp. 28
fort.13	profile_in.dat	text input qin file, read if <i>iqin = 1</i>	Inlet and Exit Profiles, pp. 29
fort.14	profile_ex.dat	text output pex file, read if <i>ipex = 1</i>	Inlet and Exit Profiles, pp. 29
fort.15	profile_out.dat	text output span file, written if <i>ispan = 1</i>	Inlet and Exit Profiles, pp. 29

Table 4. Files used by SWIFT.

The namelist input file for SWIFT is read from fort.5 (standard input.) Printed output from SWIFT is written to fort.6 (standard output.) Files linked to other Fortran units may be used in the execution of SWIFT, depending on input options. The files are described in Table 3 above.

All input text files are read using unformatted read statements, i.e., `read(5,*)`, so you don't have to worry about getting the data in the right columns.

If `iopen = 0` (default) the files are not explicitly opened in the code. You must link the files to the appropriate Fortran unit manually, e.g.,

```
ln grid.xyz      fort.1
ln qin.q fort.2
etc.
```

If *iopen* = 1 all files are opened using the default names in Table 4. This may be most useful under Windows.

**Note:** It is also possible to input your own file names using the namelist input. Edit subroutine *openfile.f*, uncomment the one line that reads namelist 7, and recompile.

```
!      read (5,nl7)
```

Add namelist & nl7 to your input file and reset the prefix of any default file names using character strings, e.g.,

```
&nl7 grid='gold.xyz' q_in='gold.0050.q' &end
```

Any file names not reset retain their default names.

### **Running SWIFT**

First set an environment variable to the number of processors you want to use:

```
setenv OMP_NUM_THREADS n
```

where *n* is the number of processors with shared memory. For a quad-core processor use *n* = 4.

SWIFT is run as a standard Linux process:

```
swift < input_file > output_file &
```

### **Linux Shell Scripts**

Linux c-shell scripts are included for running each case. The scripts do the following:

Set prefixes *pin* and *pout* for input and output file names. Input files will be named *pin.q* and *pin.kw*. Output files will be named *pout.out*, *pout.q*, and *pout.kw*. You may want to include the iteration count in the prefixes. If you use the same prefix for both, the output files will overwrite the input files at completion.

```
set pin=input_prefix      #set input prefix here
set pout=output_prefix    #set output prefix here
```

Link the input and output files to the appropriate Fortran unit numbers. Link the grid and index files here.

```
ln grid.xyz      fort.1  #set grid file here
ln case.ind      fort.10 #set index file here
```

```
ln $qin          fort.2  #restart q input
etc.
```

Link the *k- $\omega$*  turbulence model files.

```
set kw=1          #set to zero if not running the k-w model
```

```
...
```

```
endif
```

Cat (concatenate) the namelist data below to a file called *input* until the line labeled *EOIN* is reached. Change your SWIFT input here.

```
cat > input << EOIN
  'Title goes here'
&n12 cfl=5.6 ... &end
```

```
...
```

```
EOIN
```

Run SWIFT in the background and use *tail* to follow the output.

```
swift < input > $out &
tail -f $out
```

You can kill the *tail* command with <control> c.

## SWIFT Input

Namelist input is used for most variables. Many variables have defaults assigned and can be defaulted (not input.) Defaults are given in angle brackets, <Default=value> or <default>. If no default is given the value MUST be input.

### Title

*ititle* A text string of 80 characters or less enclosed in single quotes. The text is printed to the output.

### &nam2 - Algorithm Parameters

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

*ndis* Number of evaluations of artificial viscosity per stage. More than one evaluation usually improves robustness but increases CPU time, <default = 1>.

*ndis* > 1 gives 2 evaluations at stages 1 and 2 for *nstg* = 4.

*ndis* > 1 gives 3 evaluations at stages 1, 3, and 5 for *nstg* = 5.

*icdup* Flag for the type of differencing scheme.

= 0 Central-difference schemes, requires *avisc2* and *avisc4*, <default>.

= 1 AUSM<sup>+</sup> scheme, requires *ausmk*, *refmr* and/or *refms*.

= 2 H-CUSP scheme, requires *hCUSP*, *refmr* and/or *refms*.

*cfl* Courant number, typically 5.6 (see *Multistage Runge-Kutta Scheme*, pp. 4.) 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. <default = 5.0>.

*avisc1* First-order artificial dissipation coefficient. Not recommended, but can sometimes be used to stabilize a solution that blows up at startup. Set *avisc1* = 1.0 for the first 50 – 100 iterations if necessary, but be sure to set *avisc1* = 0.0 as soon as the solution is running stably.

*avisc2* Second-order artificial dissipation coefficient. Typically 0.0 – 2.0. Use 0.0 for purely subsonic flow or 1.0 for flows with shocks, <default = 1.0>.

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

*irs* Implicit residual smoothing flag. Usually = 1. (See *Implicit Residual Smoothing*, pp. 5.)

= 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. Rarely used.

*eps* Overall implicit smoothing coefficient based on the 1-D stability limit, (see *Implicit Residual Smoothing*, pp. 5). SWIFT 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. 5). Rarely used, <default = 1.0>.

<i>itmax</i>	Number of iterations, typically 100 – 1000 per run, but 1000 – 3000 will be needed for a converged solution.
<i>ivdt</i>	Variable time step flag. = 0 Spatially constant time step. = 1 Spatially variable time step, <default, highly recommended>.
<i>ipc</i>	Preconditioning flag, (see <i>Preconditioning</i> , pp. 6). = 0 No preconditioning, <default>. = 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. Rarely used.
<i>refms, refmr</i>	Reference relative Mach numbers $M'_{ref}$ used for the preconditioning, H-CUSP and AUSM <sup>+</sup> schemes. <i>Refms</i> is an absolute Mach number used for stators and <i>refmr</i> is a relative Mach number used for rotors. Should be approximately the largest Mach number expected in the flow, but less than 1.0. Reference Mach numbers for the preconditioning, H-CUSP and AUSM <sup>+</sup> schemes are adjusted using <i>pck</i> , <i>hcuspk</i> , and <i>ausmk</i> .
<i>pck</i>	Constant used to scale $M'_{ref}$ for preconditioning (Turkel's parameter <i>k</i> .) The denominator in the preconditioning matrix is limited to be $> pck \times (M'_{ref})^2$ . Typically 0.1 – 0.3. Smaller values may improve convergence, but larger values may be necessary for stability. <default = 0.15>.
<i>hcuspk</i>	Constant used to scale $M'_{ref}$ for the H-CUSP scheme. In the H-CUSP scheme the low-speed dissipation is scaled by $\max(M', hcuspk \times (M'_{ref})^2)$ , so that <i>hcuspk</i> sets the minimum value of dissipation. Typical values are 0.05 – 0.10. Smaller values may cause wiggles in the solution. Larger values may improve convergence but will increase predicted losses. <default = 0.05>
<i>ausmk</i>	Constant used to scale $M'_{ref}$ for the AUSM <sup>+</sup> scheme. In the AUSM <sup>+</sup> scheme the numerical speed of sound is used to calculate the pressure fluxes and the pressure diffusion term. The numerical speed of sound is a function of a reference Mach number $ausmk \times (M'_{ref})^2$ , so <i>ausmk</i> also controls the dissipation of the scheme, but in a less obvious way than <i>hcuspk</i> . Typical values are 0.3 – 0.8. Larger values are needed for convergence but don't hurt accuracy. <default = 0.8>.

### **&nam3 - Boundary Condition & Code Control**

The flow equations in SWIFT are formulated using Cartesian velocity components ( $u, v, w$ ). The velocity components used in the boundary conditions depend on the geometry. Cartesian velocity components ( $u, v, w$ ) are used for linear geometries (*igeom* = 0), and modified cylindrical velocity components ( $v_m, v_\theta, v_r$ ) are used for cylindrical geometries (*igeom* = 1, the default). Here  $v_m = \sqrt{v_z^2 + v_r^2}$  is the meridional velocity component. Input variables described below are written using cylindrical components, but should be replaced with Cartesian components for linear problems.

### Inlet boundary

$P_0$  and  $T_0$  are held constant at the inlet boundary. Three flags, *ibcinu*, *ibcinv*, and *ibcinw*, determine how the inlet velocity components are determined. A single flag, *ibcin*, can be used to set some of the most commonly used combinations. Properties that are held constant are either generated from the initial condition data in the input file, or are read directly from a qin-file.

- ibcinu*            Inlet boundary condition flag for  $v_m$  .
- = 1 Extrapolate a Riemann invariant based on  $v_m$  to the inlet. Used for most problems. <default>
  - = 2 Extrapolate  $v_m$  to the inlet. Recommended for low speed flows, especially with preconditioning.
- ibcinv*            Inlet boundary condition flag for  $v_\theta$  .
- = 1  $v_\theta$  is held constant, <default>.
  - = 2  $\tan \alpha = v_\theta / v_m$  is held constant.
- ibcinw*            Inlet boundary condition flag for  $v_r$  .
- = 1  $v_r$  is held constant, <default>.
  - = 2  $\tan \phi = v_r / v_m$  is held constant.
  - = 3  $v_m$  is held tangent to the meridional grid lines at the inlet, <default>.
- ibcin*              Obsolete inlet boundary condition flag. *ibcinu* is set as above, and *ibcinv* = 2.
- = 0 or defaulted: *ibcinu*, *ibcinv* and *ibcinw* set as described above.
  - = 1 Sets *ibcinw* = 3.
  - = 2 Supersonic **meridional** inflow velocity - all quantities are held constant. (Rarely used except for the NASA supersonic throughflow fan project).
  - = 3 Sets *ibcinw* = 2.
  - = 4 Sets *ibcinw* = 1.

### Exit Boundary

Four primitive variables are extrapolated to the exit. The input parameter *prat* gives the exit pressure. The parameter *ipex* determines where *prat* is specified and determines how the spanwise pressure distribution is calculated.

- ibcex*            Exit boundary condition flag.
- = 1 *Prat* is specified as a constant. Only applicable to linear geometries, or annular geometries with radial outflow.
  - = 2 Supersonic **meridional** outflow.  $P$  is extrapolated to the boundary. *Prat* is not used. (Rarely used except for the NASA supersonic throughflow fan project).
  - = 3 *Prat* is specified at the exit. The spanwise variation of  $\bar{p}$  is found by solving the radial equilibrium equation,

$$\frac{d\bar{p}}{dr} = \bar{\rho} \frac{\bar{v}_\theta^2}{r}$$

and  $\bar{p}$  is constant blade-to-blade, <default>.

= 4 *Prat* is specified at the exit. The spanwise variation of  $\bar{p}$  is found by solving the radial equilibrium equation.  $p$  is found as a perturbation about  $\bar{p}$  using a characteristic boundary condition developed by Giles.

*ipex* Flag that tells where *prat* is specified. This can affect the stability range of compressors.

If *igeom* = 0, *prat* is held constant over the exit.

= 0 *Prat* is specified at the hub, <default>.

= -1 *Prat* is specified at the tip. Use for tip-critical compressors.

= 1 Exit pex-file is read from an exit profile on fort.14, (see below).

### Inlet and Exit Profile Controls

Inlet profiles of  $P_0, v_\theta, v_r$ , and  $T_0$ , and exit profiles of  $p_{stat}$  can be specified as boundary conditions for SWIFT. For convenience, a common file format is used for both inlet and exit (see *Inlet and Exit Profiles*, pp. 29). The profiles are input as text files containing six variables at several spanwise locations. Only the variables needed at a particular boundary are used, and the other variables are ignored. The profiles are interpolated linearly along the span of the actual grid.

Inlet and exit profile files for the current solution can be written by setting variable *ispan* = 1. The output file is written to fort.15, and can be edited to extract inlet or exit profiles that can be used for subsequent calculations. In this way a multistage machine can be modeled one row at a time by using the exit profile from one blade row as the inlet profile to the next.

*ispan* Flag for writing spanwise profiles to fort.15.

= 0 No output generated, <default>.

= 1 Spanwise profile output written to fort.15.

*iqin* Flag for reading inlet profile.

= 0 Inlet conditions are calculated by subroutine qincalc based on the initial condition data, boundary layer thicknesses, etc. in the input file. Current input values are used, so the inlet profiles can be changed at restart if desired, <default>.

= 1 Inlet qin-file read from fort.13. Used to read an exit profile from a solution of an upstream blade row.

*ipex* Flag for reading exit pressure profile, also used to set location of *prat*. (see *Exit Boundary* above.)

= 1 Exit pex-file is read from fort.14.

### Code Control

*isymh* Bottom-plane symmetry flag for ducts or linear cascades.

= 1 Symmetry condition on k = 1.

else Solid wall boundary condition on k = 1, <default>.

*isymt* Top-plane symmetry flag for ducts or linear cascades.  
= 1 Symmetry condition on  $k = km$ .  
else Solid wall boundary condition on  $k = km$ , <default>.

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

*iresti* Flag for reading input restart file. Restart files are in PLOT3D format.  
= 1 Read restart file from fort.2.  
else No action taken, <default>.

*iresto* Flag for writing output restart file.  
= 1 Write restart file to fort.3, <default>.  
else No action taken.

*newkw* Flag for running the  $k-\omega$  turbulence model from scratch using a constant flow solution. Useful for debugging the  $k-\omega$  model.  
= 0 Run the  $k-\omega$  model and flow solver, <default>.  
= 1 Run the  $k-\omega$  model from initial guess for *itmax* cycles. Write the  $k-\omega$  file to fort.8 and stop.

*kwvars* Number of variables to store in the  $k-\omega$  file, (see *Turbulence Model k- $\omega$  File*, pp. 28).  
= 3 Stores 3 variables  $[\mu_{tur}, k, \omega]$ . Saves storage but not PLOT3D compatible.  
= 5 Stores 5 variables  $[\mu_{tur}, k, \omega, Re_{tur}, \mu_{lam}]$ . Increases storage, but makes the  $k-\omega$  file PLOT3D compatible, <default = 5>.

*iopen* Flag for opening input and output files explicitly by name.  
= 0 Files are read or written to Fortran units without explicitly opening them, <default>.  
= 1 Files are opened by name:  
grid.xyz = main grid file (binary)  
index.dat = SWIFT index file (text)  
etc., see *File Names*, pp. 13.

#### **&nam4 - Flow Parameters**

*igeom* Flag for linear cascade or annular blade row.  
= 0 Linear cascade.  
= 1 Annular blade row <default>.

*ga* Ratio of specific heats  $\gamma$ , <default = 1.4 for air>.

*om* Normalized blade row rotational speed,  $om = \Omega / c_{0r}$ , where  $\Omega$  is the wheel speed in radians per second, and  $c_{0r}$  has dimensions of [grid units/sec], giving *om* dimensions of [1/grid units].

Looking in the positive  $x$ -direction of the grid, clockwise rotation is negative and counterclockwise rotation is positive.  $om$  is negative for most Glenn geometries. For any new problem it is best to set  $om$ , run 1 iteration with  $oar = 1$ , then plot relative velocity vectors on a blade-to-blade plane. If the vectors go through the blade, change the sign on  $om$ . <default = 0>.

*prat* Ratio of the exit static pressure to the reference total pressure,  $prat = p_{exit} / P_{0r}$ .

*expt* Exponent used to specify the inlet whirl distribution.

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

where  $M_{\theta mid}$  is the mid-span value of  $M_{\theta}$  determined from the initial condition input.

= 0 Gives uniform  $M_{\theta}$  except within the endwall boundary layer, <default>.

= -1 Gives free vortex inflow.

= 1 Gives forced vortex inflow.

### &nam5 - Viscous Parameters

*ilt* Inviscid, Laminar, or Turbulent analysis.

= 0 Inviscid. Most other 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 but may over predict losses for transonic compressors.

= 4 Fully turbulent using the Wilcox baseline  $k-\omega$  turbulence model.

= 5 Turbulent with transition using the Wilcox low Reynolds number  $k-\omega$  turbulence model. Note that low Reynolds number model refers to the transition model, and **not** to near-wall modifications needed by  $k-\epsilon$  models.

*isst* Flag for the stress limiter in Wilcox's 2006  $k-\omega$  model. Limits the turbulent viscosity when production of turbulent kinetic energy exceeds destruction. Works well for shock separated flows.

= 0 No stress limiter, <default>.

else Stress limiter is used.

*itur* The turbulence model is updated every  $itur$  iterations. Recommended values are  $itur = 5$  for the Baldwin- Lomax or Cebeci-Smith models, and  $itur = 2$  for the  $k-\omega$  model. If the  $k-\omega$  model blows up quickly it may help to use  $itur = 1$  for the first 100 – 200 iterations, <default = 5>.

*renr* Reynolds number per unit length based on reference conditions,  $renr = \rho_{0r} c_{0r} / \mu_{0r}$ . Must have units of [1/grid units]. Generally much larger than a conventional free-stream Reynolds number. For example, for standard conditions:

$$\begin{aligned} renr &= .0765 \left( \frac{\text{lbm}}{\text{ft}^3} \right) \times 1116.7 \left( \frac{\text{ft}}{\text{sec}} \right) / 1.197 \times 10^{-5} \left( \frac{\text{lbm}}{\text{ft sec}} \right) \\ &= 7.143 \times 10^6 / \text{ft} \end{aligned}$$

<i>prnr</i>	Prandtl number, <default = 0.7 for air>.
<i>tw</i>	Normalized wall temperature, $tw = T_{wall} / T_{0r}$ . = 0 Adiabatic wall boundary conditions are used. = 1 $T_{wall} = T_{0r}$ <default>. else $T_{wall} = tw \times T_{0r}$ .
<i>vispwr</i>	Exponent for laminar viscosity power law, <default <i>vispwr</i> = 0.667 for air>. Use <i>vispwr</i> = 0.0 for water. $\mu / \mu_{0r} = (T / T_{0r})^{vispwr}$
<i>prtr</i>	Turbulent Prandtl number, <default = 0.9>.
<i>cmutm</i>	Value of $\mu_{turb} / \mu_{0r}$ where transition is assumed to occur for the Baldwin-Lomax and Cebeci-Smith models. Baldwin and Lomax recommended 14. Can be increased to move transition downstream or vice-versa. If <i>cmutm</i> = 0 the flow is fully turbulent, <default = 14>.
<i>jedge</i>	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 ( <i>ilt</i> = 2) <i>jedge</i> should be a grid line slightly bigger than the largest expected blade boundary layer. For the Cebeci-Smith turbulence model ( <i>ilt</i> = 3), <i>jedge</i> should be a grid line slightly bigger than half the largest expected blade boundary layer, <default = 10>.
<i>kedgh, kedgt</i>	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> . <default = 10>.
<i>iltin</i>	Flag controlling inlet velocity and $P_0$ profiles. = 0 Inviscid. = 1 Laminar. = 2 Turbulent using Cole's wall-wake profile, <default>.
<i>dblh, dblt</i>	Inlet hub and tip boundary layer thicknesses in grid units.
<i>xrle, xrte</i>	Axial locations where the hub starts and stops rotating. Rotational boundary conditions are applied on the hub for $xrle < x < xrte$ . Stationary conditions are applied elsewhere. Note that <i>xrle</i> and <i>xrte</i> may not be sufficient to locate the rotating part of the hub in a radial flow machine. Defaults are set to make the entire hub rotate.
<i>irle, irte</i>	i-indices where the hub starts and stops rotating. In radial flow machines <i>xrle</i> and <i>xrte</i> may not be sufficient to locate the rotating part of the hub, so <i>irle</i> and <i>irte</i> can be used instead. This option only works correctly for a single block H-grid, and even then the grid lines may not be straight. Defaults are set to make the entire hub rotate.
<i>tintens</i>	Free-stream turbulence intensity written as a decimal. Used to set the inlet value of k for the k- $\omega$ model. Typically 0.1 or less. Very small values can cause an abrupt decay of $\omega$ very close to the inlet. <default = 0.01, i.e., one percent>.
<i>tmuinf</i>	Free-stream turbulent viscosity, Used to set the inlet value of $\omega$ for the k- $\omega$ model

*tlength* Old input variable used to determine the inlet value of  $\omega$ . Retained for backwards compatibility.  
Turbulent length scale in grid units. Typically  $0.03 \times$  boundary layer height, or  $0.001 \times$  pitch.  
*Tmuinf* is used *tlength* is omitted, <default>.

*hrough* Surface roughness height in grid units. Usually *hrough* = 0 is used to model a hydraulically smooth surface. To model a rough surface like a turbine blade set *hrough* = equivalent sand grain roughness height, or 2 – 4 times the RMS roughness height. If the first grid point off the wall is at  $y^+ = 2$ , then *hrough* must be  $> 2.5 \times \Delta y_{wall}$  to have much effect.  
= 0. Hydraulically smooth surface.  
> 0. Roughness effects are modeled using the Cebeci-Chang model or Wilcox's roughness boundary condition for  $\omega$ .

### **&nam6 - Output Control**

*oar* Flag for frame of reference of output q-file. SWIFT automatically detects the frame of reference of a restart q- file and converts it to the absolute frame for internal use if necessary.  
= 0. All blade rows are in the absolute frame of reference.  
= 1. All blade rows are in the relative frame of reference. <default>

*mioe* Flag for output format of mass flow in 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>.  
= 4 Eliminates the maximum residual (which always looks like the RMS residual anyway) and prints both  $\dot{m}_{in}$  and  $\dot{m}_{out}$  instead. Calculate the mass flow error yourself with EXCEL if desired.

*iqav* Flag controlling type of  $\theta$ -averaging used in the output.  
= 0 Entropy average. Mass average of  $[s, V, h]$ . Gives a good estimate of local losses, <default>.  
= 1 Momentum average. Mass average of  $[\rho, \rho V, e]$ , fairly conservative, similar to the mixed-out average.  
= 2 Mixed-out average. Formal average of inviscid fluxes gives properties far downstream. Usually the most conservative average.  
= 3 Total pressure average. Converts  $P_0$  to an equivalent  $T_0$ , mass averages, then converts back. Often done with experimental data. Usually similar to the entropy average.

*nko* Number of k-indices for blade surface output, max = 10, <default = 0>.

*ko* Array of *nko* k-indices separated by commas where blade surface output is desired, <default = 0>.

*iog* Grid block number where spanwise output is desired. Spanwise output is normally printed at the inlet and exit of each blade grid. Sometimes it is useful to have output from other cross-channel planes, say near the trailing edge. This output can be generated for a single grid block by specifying the block number *iog*, and the desired i-indices *io*, <default = 1>.

*io* Array of up to 10 i-indices where spanwise output is desired. For H-grids output is printed at each i index. For C-grids the i-index and its periodic neighbor are merged, <default = 0>.

*ismout* Flag for normalized distance *sbar* printed in blade surface output.  
= 1 *sbar* = arc length around blade.  
= 2 *sbar* = meridional distance along the blade.  
else *sbar* = axial distance *x*, <default>.

*ileout* Flag for location of leading edge (*sbar* = 0) printed in blade surface output.  
if *ismout* = 0: LE is at *xmin*  
if *ntype* = 1: LE is at *ile* (H-grid)  
if *ileout* = 0: LE is at the middle i-index  
else: LE is at *ileout*

## Initial Condition Input

$2+nrow$  lines of data must be input immediately after the namelist input. This data is used for the initial guess and the inlet boundary conditions.  $Nrow$  is the number of blade rows. The first line is ignored and is usually used for column labels. The second line gives the inlet conditions at mid span, and is used to set the inlet boundary conditions. The remaining  $nrow$  lines give row number and nominal flow exit conditions at mid span. Unformatted reads are used; so all variables must be input.

A sample initial condition input for a seven-block grid is shown in Figure 5. A portion of the input is repeated below.

row	P0	Mx	Mt	Mr	T0
0	1.0000	.1330	-.0000	0.	1.0000
1	.9938	.1692	-.3986	0.	1.0000
etc.					

The variables are as follows:

row	Integer blade row number. Row number 0 is the inlet. Subsequent row numbers represent the exits of each blade row.
P0	$P_0 / P_{0r}$ at mid span.
Mx	Mid span Mach number in the x-direction.
Mt	Mid span Mach number in the q- or y-direction.
Mr	Mid span Mach number in the r- or z-direction.
T0	$T_0 / T_{0r}$ at mid span.

## Index File

The index file is a text file that gives the grid sizes, connectivity, and some boundary condition information for each grid block.

The first line is ignored and can be used for column labels. Subsequent lines give grid type, dimensions, key indices, connecting grid numbers, blade row number, and relative rotational rate for each grid. Negative values are sometimes used to toggle boundary condition options. One line is required for each grid. Unformatted reads are used; so all variables must be input.

For an isolated blade row, TCGRID will produce a complete index file written to fort.10. It may be necessary to modify *nhub* or *ntip* if the simple periodicity clearance model is to be used, or to modify the rotation multipliers *om*, *omh*, or *omt*.

For multistage calculations the grids and index files for each blade row are generated separately, and merged using the utility code called *multix.f*. The merged index file will have the correct block sizes and key indices, but must be edited manually to set block connectivity, and options for mixing planes, clearances, and endwall rotation. Additional details about generating multistage grids are given in the TCGRID user's manual [16].

A sample index file for a seven-block grid is shown in Figure 5. A portion of the file is repeated below.

```

grid type   im   jm   km   i1   i2   i3   nin   nex   nhub   ntip   nlr   row   om   omh   omt
  1     1    17    16    57    0    0    999    2     0     0     0     1   0.   1.   0.
  2     2   147    37    57    24   67    0     1    -3     0     0     1   0.   1.   0.
  3     2   161    43    57    27   74    0    -2    -5     0     4     2   1.   1.   0.
etc.

```

## Index File Variables

grid	Grid (block) number, from 1 to number of grids.
type	Flag giving type of grid. = 1 H grid for upstream = 2 C grid for blades = 3 O grid for hub or tip clearances
im	Number of grid points in i-direction.
jm	Number of grid points in j-direction.
km	Number of grid points in k-direction.
i1 (C-grid)	Lower i-index of the trailing edge. Upper index is assumed to be periodic.
i1 (H-grid)	Leading-edge index of an H-grid, or flag for the type of H-grid geometry. = 0 Upstream H-grid ahead of a C-grid = 1 H-grid in a rectangular duct > 1 H-grid around a blade. i1 is the i-index of the leading edge.
i2 (C-grid)	Lower i-index of the inlet. Upper index is assumed to be periodic.
i2 (H-grid)	i-index of the trailing edge for an H-grid around a blade
i3	Unused, set to 0.
nin	Inlet boundary condition flag.

	<p>= 999 C- or H-grid with conventional inlet boundary condition.</p> <p>&gt; 0 C-grid inlet patched to upstream H-grid number nin.</p> <p>&lt; 0 C-grid inlet mixed-out from upstream C-grid number nin.</p>
nex	<p>Exit boundary condition flag.</p> <p>= 999 C-grid with conventional exit boundary condition.</p> <p>&gt; 0 H-grid exit patched to downstream C-grid number nex.</p> <p>&lt; 0 C-grid exit mixed out to downstream C-grid number nex.</p>
nhub	<p>Flag for hub clearance.</p> <p>&gt; 0 Connecting grid block number for gridded hub clearance.</p> <p>= 0 No hub clearance.</p> <p>&lt; 0 Simple periodicity hub clearance model between <math>k=1</math> and <math>k=\text{abs}(\text{nhub})</math>.</p>
ntip	<p>Flag for tip clearance.</p> <p>&gt; 0 Connecting grid block number for gridded tip clearance.</p> <p>= 0 No tip clearance.</p> <p>&lt; 0 Simple periodicity tip clearance model between <math>k=\text{abs}(\text{ntip})</math> and <math>k=\text{km}</math>.</p>
nlr	Unused, set to 0.
row	Integer blade row number between 1 and the number of blade rows. Corresponds to row number in initial condition input.
om(n)	Rotation multiplier. The rotational speed for row n is $om \times om(n)$ . Usually 0.0 for stators, 1.0 for rotors, or -1.0 for counter-rotating rotors. (See <i>&amp;nam5 – Viscous Parameters</i> , pp. 20, for definition of normalized blade row rotational speed <i>om</i> ).
omh(n)	Hub rotation multiplier. Rotational speed for $k = 1$ for row n is $om \times omh(n)$ . Usually 1.0 for rotating hubs. Overridden by variables <i>xrle</i> and <i>xrte</i> , or <i>irle</i> and <i>irte</i> , the axial locations or grid indices where the hub starts and stops rotating (see <i>&amp;nam5 – Viscous Parameters</i> , pp. 20).
omt(n)	Tip rotation multiplier. Rotational speed for $k = \text{km}$ for row n is $om \times omt(n)$ . Usually 0.0 for stationary shrouds or 1.0 for rotating shrouds.

## SWIFT Output

Printed output from SWIFT is written to fort.6 (standard output,) and contains the following information:

- The input variables are echoed back for reference, and any comments or warnings about the input are given.
- Spanwise profiles of  $\theta$ -averaged flow variables are given at the inlet or exit. These variables come from the initial guess if *iresti* = 0, or from the restart file if *iresti* = 1. The initial profiles are often useful for identifying grid indices near the tip clearance, or near the edge of endwall boundary layers.
- A convergence history that gives maximum and RMS residuals of density, and exit flow properties versus iteration.
- Total CPU time for all processors.

- Spanwise profiles of  $\theta$ -averaged flow variables are given at the inlet and exit for the new solution. Four different averaging schemes are available for computing these profiles. Most variables should be obvious, except:

*% mdot* is the percent of mass flow between the hub and index *k*.

The last value in *% mdot* is the total mass flow through the annulus at that station,  $\dot{m} / (\rho_{0r} c_{0r})$ .

- Spanwise profiles of blade row performance parameters are given after each blade row. Most variables should be obvious, except:

$$\alpha = \tan^{-1}(v_{\theta} / v_m)$$

$$\phi = \tan^{-1}(v_r / v_m)$$

- Blade surface distributions are given of the following quantities:

*x* x-coordinate, grid units

*r* r-coordinate, grid units

*sbar* normalized arc length from the leading edge.

*ps*  $p / P_{0ref}$

*ts*  $T / T_{0ref}$

*M\_isen* Isentropic Mach number

*M\_rel* Relative Mach number, usually 0.0 except for inviscid solutions.

*y+* Grid spacing at the wall in turbulent wall units. Should < 2 – 3 at most grid points for accurate prediction of losses and heat transfer.

1000\*Cf Skin friction coefficient

$$C_f = \mu \left. \frac{\partial V'}{\partial n} \right|_{wall} / \left( \frac{1}{2} \rho_{in} V_{in}'^2 \right)$$

1000\*St Stanton number, usually 0.0 if *tw* = 0.

$$St = k \left. \frac{\partial T}{\partial n} \right|_{wall} / \left[ \rho_{in} V_{in}' C_p (T_{in} - T_{wall}) \right]$$

*tmu\_max* Maximum values of  $\mu_{turb} / \mu_{0r}$  along each *i* grid line. Can be used to identify transition. For the Baldwin-Lomax and Cebeci-Smith models transition occurs where *tmu\_max* jumps abruptly from 0 to > *cmutm*. For the *k*- $\omega$  model transition occurs more gradually but should be obvious as a rapid growth of *tmu\_max*.

## File Descriptions

Code for reading PLOT3D files given below is for single-block grids only. Consult the PLOT3D documentation for details on how to read multi-block files.

### Grid XYZ-File

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

```
c      read q-file
      read(2) im, jm, km
      read(2) eminf, aldeg, renr, time
      read(2) (((q(1, 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)
```

The q-variables are:

$$q = \left[ \frac{\rho}{\rho_{0r}}, \frac{\rho u}{\rho_{0r} c_{0r}}, \frac{\rho v}{\rho_{0r} c_{0r}}, \frac{\rho w}{\rho_{0r} c_{0r}}, \frac{e}{\rho_{0r} c_{0r}^2} \right]$$
$$e = \rho \left[ C_v T + \frac{1}{2} (u^2 + v^2 + w^2) \right]$$

If  $oar = 1$  the relative velocity components are stored,  $v' = v - \Omega z$ ,  $w' = w + \Omega y$ .

### Turbulence Model k- $\omega$ File

Restart files for the k- $\omega$  turbulence model are stored in standard PLOT3D q-file format. Solution files can be read with the following Fortran code:

```
c      read tmu, k, w
      read(7) im, jm, km
      read(7) dum
      read(7) (((tkw(1, i, j, k), i=1, im), j=1, jm), k=1, km), l=1, kwvars)
```

The tkw-variables are:

$$tkw = \left[ \frac{\mu_{tur}}{\mu_{0r}}, \frac{k}{c_{0r}^2}, \frac{\omega}{c_{0r}}, \frac{\mu_{lam}}{\mu_{0r}}, Re_{tur} \right]$$

Note that the laminar viscosity  $\mu_{lam}$  and the turbulence Reynolds number  $Re_{tur}$  are not used by SWIFT. They are written to pad the file for PLOT3D compatibility if  $kwvars = 5$ . This results in larger file sizes than necessary. Smaller files may be generated by setting  $kwvars = 3$ , but the files cannot be read by PLOT3D.

### **Inlet and Exit Profiles**

Inlet profiles of  $P_0$ ,  $v_x$ ,  $v_\theta$ ,  $v_r$ , and  $T_0$ , and exit profiles of  $p_{stat}$  can be specified as boundary conditions for SWIFT. For convenience, a common file format is used for both inlet and exit. The profiles are input as text files containing six variables at several spanwise locations. Only the variables needed at a particular boundary are used, and the other variables are ignored. For example,  $v_x$  is ignored at the inlet and can be set to zero. The profiles are interpolated linearly along the span of the actual grid.

A sample profile file can be generated by setting variable  $ispan = 1$ . The output written to fort.15 may be edited manually to extract the desired profile. The format is as follows:

```
&ospan irow = 0 kin = 95 flow = 119.25082 &end
k      s/span      P0/P0i      vx/c0      vth/c0      vr/c0      T0/T0i      ps/P0i
1      0.00000      1.12907      0.00000      -0.65789      0.00000      1.06326      0.83876
2      0.00019      0.90181      -0.07050      -0.31211      -0.01668      1.00000      0.83864
etc.
```

The first line is namelist input. Only  $kin$  is required.

kin	Number of spanwise points.
irow	Dummy variable not used by SWIFT, but useful for identifying the desired profile from an output file. <i>Irow</i> gives the location of the profile, where $irow = 0$ is the inlet, $irow = 1$ is the exit of the first blade row, $irow = 2$ is the exit of the second blade row, etc.
flow	Dummy variable not used by SWIFT. <i>Flow</i> is the non-dimensional mass flow and is included for use by the CSTALL code now under development.

The second line has titles for convenience but is not read. The remaining  $kin$  lines have the following variables:

k	Spanwise index, not used.
s/span	Normalized spanwise distance, between 0.0 at the hub to 1.0 at the tip.
P0/P0i	Normalized total pressure, used for inlet profiles only.
vx/c0	Normalized axial velocity, not used.
vth/c0	Normalized tangential velocity, used for inlet profiles only.
vr/c0	Normalized radial velocity, used for inlet profiles only.
ps/p0i	Normalized static pressure, used for exit profiles only.

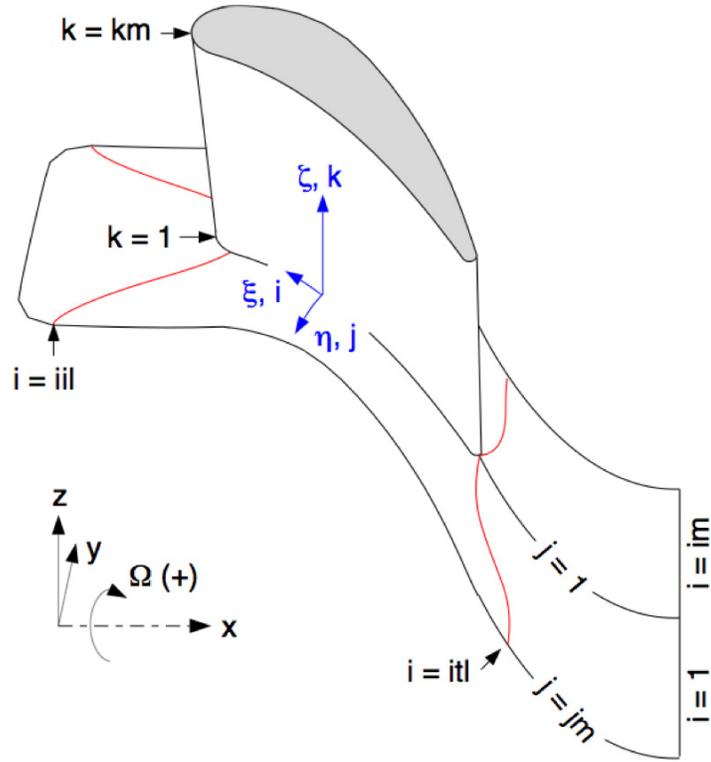


Figure 1. 3-D coordinate system and grid index convention.

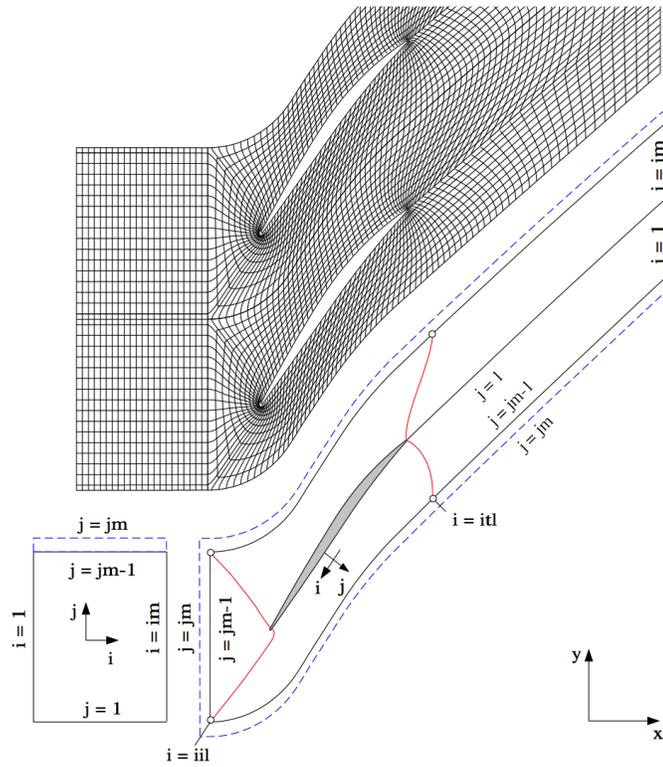


Figure 2. Index convention for a C-grid around a fan blade with an H-grid upstream.

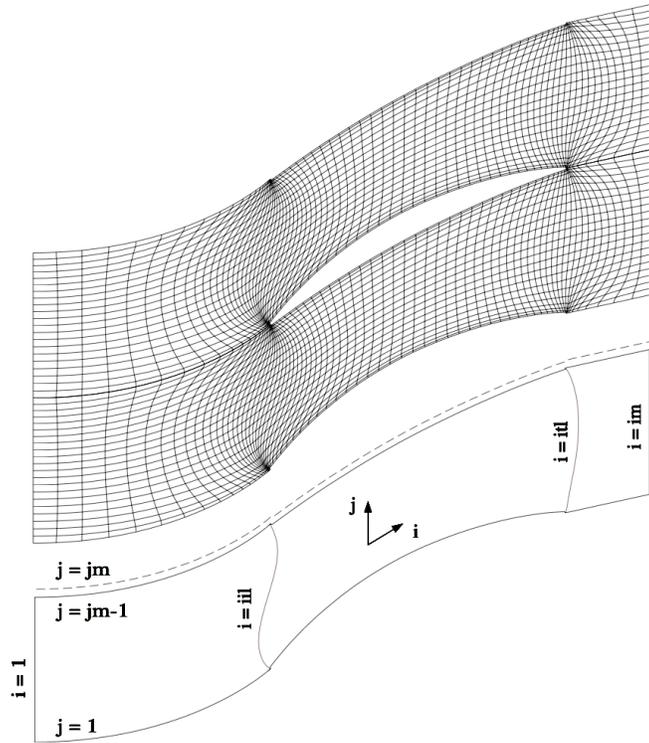


Figure 3 Index convention for an H-grid around a fan blade.

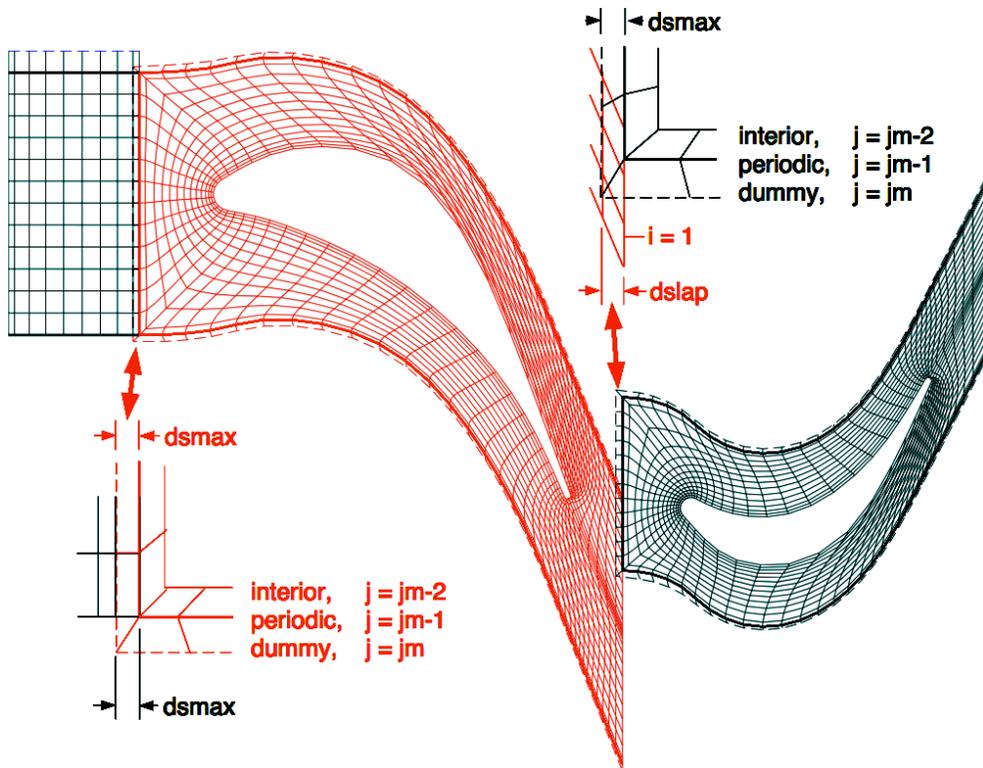
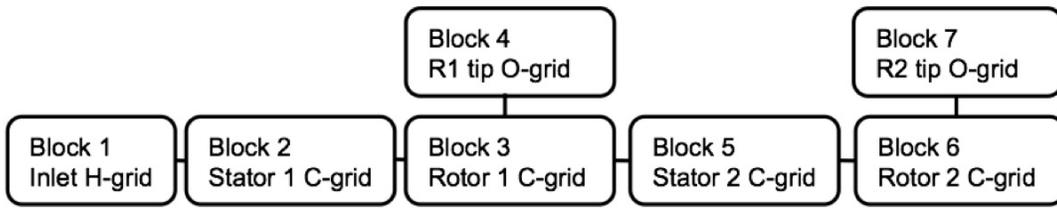
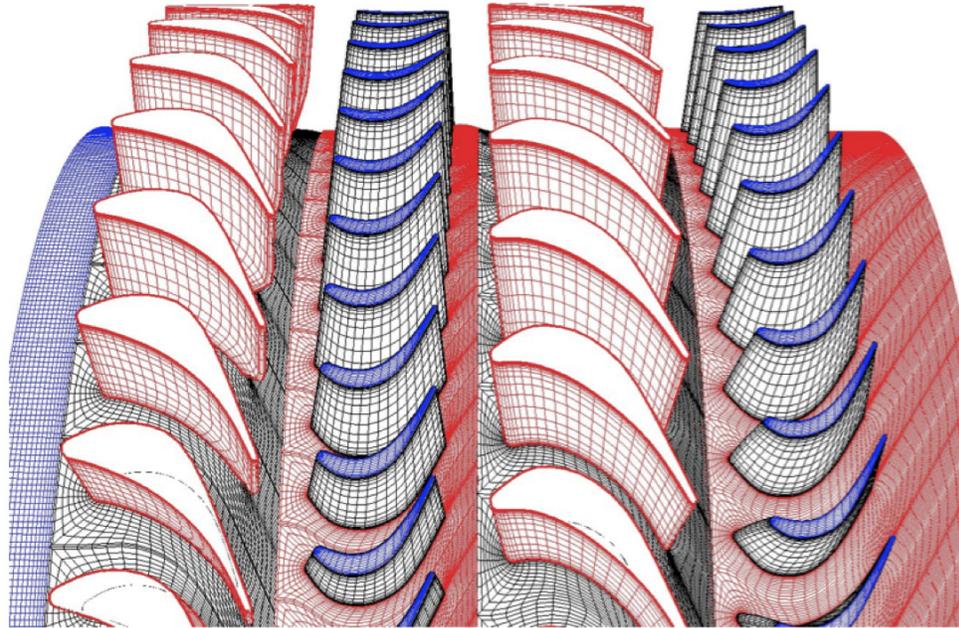


Figure 4. Three-block grid for a turbine stage showing overlap regions and dummy grid lines.



Top: Seven-block grid for the SSME two-stage fuel turbine with rotor tip clearances.  
 Bottom: Block diagram of the SSME seven-block grid.

grid	type	im	jm	km	i1	i2	i3	nin	nex	nhub	ntip	nlr	row	om	omh	omt
1	1	17	16	57	0	0	0	999	2	0	0	0	1	0.	1.	0.
2	2	147	37	57	24	67	0	1	-3	0	0	0	1	0.	1.	0.
3	2	161	43	57	27	74	0	-2	-5	0	4	0	2	1.	1.	0.
4	3	109	13	13	7	49	0	0	0	0	3	0	2	1.	1.	0.
5	2	147	37	57	24	65	0	-3	-6	0	0	0	3	0.	1.	0.
6	2	161	43	57	31	74	0	-5	999	0	7	0	4	1.	1.	0.
7	3	101	13	13	7	43	0	0	0	0	6	0	4	1.	1.	0.

Index file on fort.10

row	P0	Mx	Mt	Mr	T0
0	1.0000	.1330	-.0000	0.	1.0000
1	.9938	.1692	-.3986	0.	1.0000
2	.8210	.1984	.0802	0.	.9518
3	.8112	.1858	-.4175	0.	.9518
4	.7964	.3693	.0852	0.	.9059

Initial condition data in SWIFT input.

Figure 5. Grid, block diagram, index file, and initial condition data for the SSME two-stage fuel turbine.

## Test Cases

### Goldman's Annular Turbine Vane Cascade

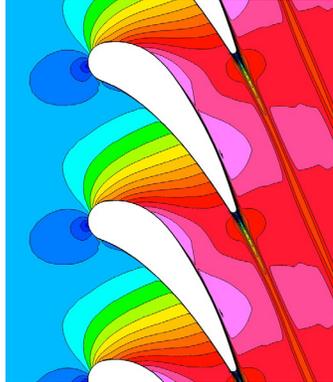


Figure 6. Mach contours for the Goldman annular turbine vane cascade.

This test case is an annular turbine vane cascade described by Goldman and McLallin in [17], with computational results shown in [1, 4]. The blade is a simple extruded section with constant radius endwalls. There are 36 vanes with a design exit Mach number of 0.665. Figure 6 shows Mach number contours through the cascade at mid span.

The c-shell script `gold.csh` explains how to run the case with the central-difference, AUSM<sup>+</sup>, and H-CUSP schemes. The Excel spreadsheet `Goldman_cascade_data.xlsx` compares blade surface pressures and exit total pressure loss coefficients to experimental data. Experimental wake profiles are also included.

### Space Shuttle Main Engine Two-Stage Fuel Turbine

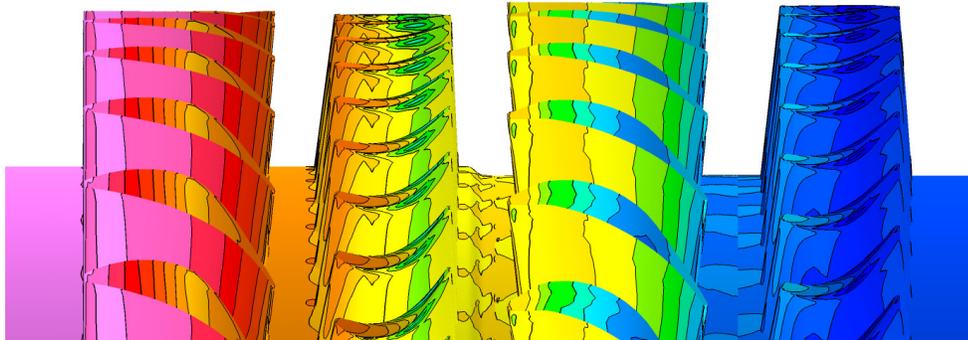


Figure 7. Pressure contours on the space shuttle main engine fuel turbine.

The space shuttle main engine (SSME) used turbopumps to pump fuel and oxidizer from the main tank to the combustion chamber. The high-pressure fuel turbopump used a two-stage axial flow turbine to drive the pump. Hudson, et al tested the turbine experimentally in a cold flow test at NASA Marshall Space Flight Center, and measured surface pressures on the stators and endwalls [19]. Dunn, et al. tested the turbine in a short duration shock tube at Calspan, and measured blade heat transfer and unsteady pressures [18]. The SSME turbine was used as a test case for the mixing plane capability in SWIFT in [15].

The test case uses a seven-block grid that is generated by running the script named `makegrid.csh`. The index file `out.ind` must be modified manually to set the connectivity at the mixing planes, but the final index file, `ssme.ind`, is included. The script `ssme.csh` runs SWIFT 2500 iterations with the AUSM<sup>+</sup> scheme, the  $k-\omega$  turbulence model, and preconditioning. Preconditioning significantly improves convergence for this case even though the flow speed is relatively high.  $tw$  is set to 0.7 for heat transfer calculations. The Excel spreadsheet, `ssme_data.xlsx`, compares measured and computed pressures on the stators, and Stanton numbers on the stators and first rotor.

## NASA Large, Low-Speed Centrifugal Compressor

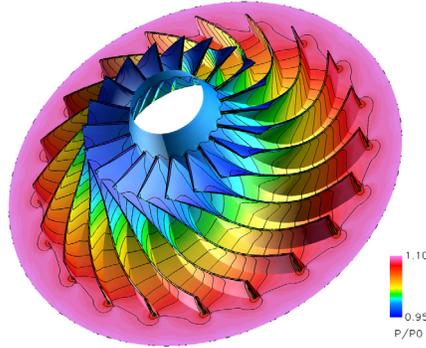


Figure 8. Pressure contours on the large, low-speed centrifugal compressor.

The large, low-speed centrifugal compressor (LSCC) is a research rig used to make detailed measurements in a centrifugal compressor [20]. It is 5 feet in diameter and has 20 blades with no splitters. It was used in [14] as a test case to demonstrate preconditioning in SWIFT. Computed pressure contours on the impeller are shown in Figure 8.

The test case uses a single-block H-grid. Running TCGRID with the input file `lsc.c` generates the grid. The index file `lsc.ind` sets up the periodic tip clearance model over the rotor. The script `lsc.csh` runs SWIFT with the AUSM<sup>+</sup> scheme, the  $k-\omega$  turbulence model, and preconditioning. The Excel spreadsheet, `lsc-data.xlsx`, compares measured and computed blade surface pressures and exit total pressure profiles.

### Rotor 37

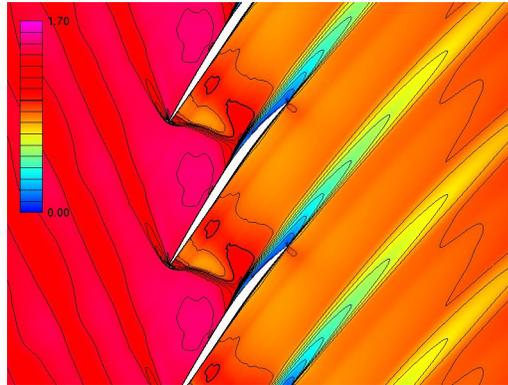


Figure 9. Relative Mach number contours for rotor 37 at mid span.

Rotor 37 is a low aspect ratio inlet rotor for a core compressor. It has 36 multiple circular-arc (MCA) blades and a design pressure ratio of 2.106 at a mass flow of 44.5 lb.sec. It was originally tested as a stage by Reid and Moore [22, 23], and later the isolated rotor was tested by Suder, et al. [24]. Suder's measurements were used for a blind CFD test case by IGTI and AGARD. SWIFT calculations for rotor 37 were shown in [24], and computed Mach contours at mid span are shown in Figure 9.

The 3-block grid for the rotor 37 test case is generated by running TCGRID with the input file `r37.int`. The index file generated by TCGRID should be identical to the included index file `r37.ind`. SWIFT input `r37.csh` is set up to run 3000 iterations with the AUSM<sup>+</sup> scheme, the  $k-\omega$  turbulence model. The exit static pressure ratio  $pr_{at}$  is set to give a solution near peak efficiency. The Excel spreadsheet `rotor_37.xlsx` includes a speed line computed with other pressure ratios. It also compares measured and computed profiles of pressure ratio, temperature ratio, adiabatic efficiency, and flow angle behind the rotor.

### Stage 35

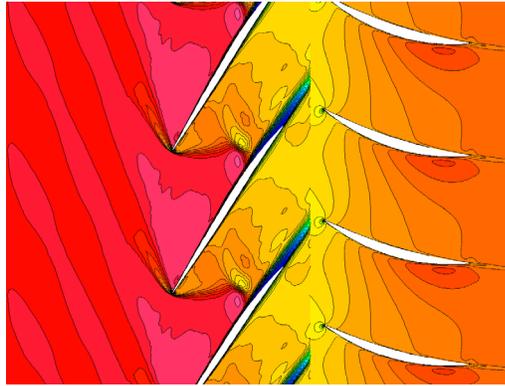


Figure 10. Relative Mach number contours for stage 35 at mid span.

Stage 35 is an inlet stage for a core compressor tested by Reid and Moore [22, 23, 25]. SWIFT calculations for stage 35 were also shown in [24]. Computed Mach contours at mid span are shown in Figure 10.

Rotors 35 and 37 have the same blade count, design speed, hub and casing radii, and tip clearance. Rotor 35 has the same blade profile as rotor 37 in the front, transonic half of the blade, so that the shock structure and choking flow of the two rotors are the same. However, rotor 37 has more camber than rotor 35 aft of the shock, giving a design pressure ratio of 2.106, while stage 35 has design pressure ratios of 1.865 for the rotor and 1.82 for the stage. Stator 35 has 46 MCA blades cantilevered from the casing with a clearance of  $\sim 0.5$  percent span.

The 5-block grid for the stage 35 test case is generated by running the script `makegrid.csh`. The index file `out.ind` must be modified manually to set the connectivity and other parameters, but the final index file, `stage35.index`, is included. The script `stage.csh` runs SWIFT 3000 iterations with the AUSM<sup>+</sup> scheme and the  $k-\omega$  turbulence model. The exit static pressure ratio  $pr_{at}$  is set to give a solution near peak efficiency. The Excel spreadsheet `stage_35.xlsx` includes a speed line computed with other pressure ratios, and compares spanwise profiles of several quantities behind the rotor and stator.

### Rotor 67

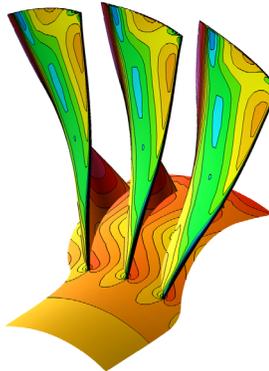


Figure 11. Surface pressure contours on rotor 67.

Rotor 67 is the first stage rotor of a two-stage fan, with a design pressure ratio of 1.63 at a mass flow of 73.3 lb/sec. The rotor has 22 blades. Rotor 67 was tested by Strazisar, and the results were presented as a CFD test case in Fottner [25]. CFD solutions for rotor 67 were computed with the RVC3D code and were presented in [2].

SWIFT solutions for rotor 67 are computed on a 3-block grid that is generated by running TCGRID with the `r67.int` input file. The index file generated by TCGRID should be identical to the included index file `r67.ind`. SWIFT input in the script `r67.csh` is set up to run 2000 iterations with the central-difference scheme and the  $k-\omega$  turbulence model. The Excel spreadsheet `rotor_67.xlsx` compares measured and computed speed lines for total pressure, and profiles of pressure ratio, temperature ratio, adiabatic efficiency, and flow angle behind the rotor.

## 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. Liou, M.-S., and Steffen Jr. C. J., "A New Flux Splitting Scheme," *J. Computational Physics*, Vol. 107, No. 1, July 1993, pp 23-29.
4. Chima, R. V., and Liou, M.-S., "Comparison of the AUSM<sup>+</sup> and H-CUSP Schemes for Turbomachinery Applications," AIAA Paper 2003-4120. Also NASA TM-2003-212457.
5. Tatsumi, S., Martinelli, L., and Jameson, A., "Design, Implementation, and Validation of Flux Limited Schemes for the Solution of the Compressible Navier-Stokes Equations," AIAA Paper 94-0647, Jan. 1994.
6. Tatsumi, S., Martinelli, L., and Jameson, A., "A New High Resolution Scheme for Compressible Viscous Flow with Shocks," AIAA Paper 95-0466, Jan. 1995.
7. Baldwin, B. S., and Lomax, H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257, Jan. 1978.
8. 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.
9. Cebeci, T. and Chang, K. C., "Calculation of Incompressible Rough-Wall Boundary Layer Flows," *AIAA Journal*, Vol. 16, July, 1978, pp. 730-735.
10. Wilcox, D. C., Turbulence Modeling for CFD, Third Edition, DCW Industries, Inc. La Canada, CA, 2006.
11. Chima, R. V., "A  $k-\omega$  Turbulence Model for Quasi-Three-Dimensional Turbomachinery Flows," AIAA Paper 96-0248, Jan. 1996. Also NASA TM-107051.
12. 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.
13. Turkel, E., "A Review of Preconditioning Methods for Fluid Dynamics," *Applied Numerical Mathematics*, Vol. 12, 1993, pp. 257-284.
14. Tweedt, D. L., Chima, R. V., and Turkel, E., "Preconditioning for Numerical Simulation of Low Mach Number Three-Dimensional Viscous Turbomachinery Flows," AIAA Paper 97-1828, June, 1997. Also NASA TM- 113120.
15. Chima, R. V., "Calculation of Multistage Turbomachinery Using Steady Characteristic Boundary Conditions," AIAA Paper 98-0968. Also NASA TM-1998-206613.
16. Chima, R. V., "TCGRID 3-D Grid Generator for Turbomachinery - User's Manual and Documentation, Version 400," July 2011. [http://www.grc.nasa.gov/WWW/5810/rvc/docs/tcgrid\\_400.pdf](http://www.grc.nasa.gov/WWW/5810/rvc/docs/tcgrid_400.pdf)
17. Goldman, L. J., and McLallin, K. L. "Cold-Air Annular Cascade Investigation of Aerodynamic Performance of Core-Engine-Cooled Turbine Vanes. I: Solid Vane Performance and Facility Description," NASA TMX-3224, 1975.
18. Dunn, M. G., Kim, J., Civinskas, K. C., and Boyle, R. J., "Time-Averaged Heat Transfer and Pressure Measurements and Comparison with Prediction for a Two-Stage Turbine," *J. Turbomachinery*, Vol. 116, Jan. 1994, pp. 14-22.
19. Hudson, S. T., Gaddis, S. W., Johnson, P. D., and Boynton, J. L., "Cold Flow Testing of the Space Shuttle Main Engine High Pressure Fuel Turbine," AIAA Paper 91-2503, June 1991.

20. Hathaway, M. D., Chriss, R. M., Strazisar, A. J., and Wood, J. R., "Laser Anemometer Measurements of the Three-Dimensional Rotor Flow Field in the NASA Low-Speed Centrifugal Compressor," NASA Technical Paper 3527, June, 1995.
21. Chima, R. V., "SWIFT Code Assessment for Two Similar Transonic Compressors," AIAA-2009-1058, Jan. 2009. Also NASA TM-2009-215520.
22. Reid, L. and Moore, R. D., "Design and Overall Performance of Four Highly-Loaded, High Speed Inlet Stages for an Advanced, High Pressure Ratio Core Compressor," NASA TP-1337, 1978.
23. Reid, L. and Moore, R. D., "Experimental Study of Low Aspect Ratio Compressor Blading," ASME Paper 80-GT-6, Mar. 1980.
24. Suder, K. L. and Celestina, M. L., "Experimental and Computational Investigation of the Tip Clearance Flow in a Transonic Axial Compressor Rotor," NASA TM-106711, 1994.
25. Reid, L. and Moore, R. D., "Performance of a Single-Stage Axial-Flow Transonic Compressor with Rotor and Stator Aspect Ratios of 1.19 and 1.26 Respectively, and with Design Pressure Ratio of 1.82," NASA TP-1228, 1978.
26. Fottner, L. ed., Test Cases for Computation of Internal Flows in Aero Engine Components, chapter V1.2, Test Case E/CO-2 Single Transonic Fan Rotor, by Wood, J. R., Strazisar, T., and Hathaway, M., AGARD Advisory Report No. 275, July, 1990.