Design of a High Pressure Ratio Fan Stage to Take Advantage of Boundary Layer Suction

by

Lawrence M. Smilg

S.B., Massachusetts Institute of Technology (1993)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1994

Author . . . .
Department of Aeronautics and Astronautics
August 22, 1994

Certified by . . . .
Professor Jack Kerrebrock
Richard Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by . . . .
Professor Harold Y. Wachman
Chairman, Departmental Committee on Graduate Students

Design of a High Pressure Ratio Fan Stage to Take Advantage of Boundary Layer Suction

by

Lawrence M. Smilg

Submitted to the Department of Aeronautics and Astronautics
on August 22, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis presents a design for a high pressure fan stage suitable for use as the first stage of a commercial next generation high-bypass ratio turbofan engine. The motivation for a high pressure ratio fan stage is to optimize propulsive efficiency by matching fan and core exit velocities for the turbofan engine. The high pressure ratio of the stage is made possible by using suction along the chord of the blade to delay boundary layer separation. The design was made by using a streamline curvature program, SC, to compute the fan throughflow, then using MISES, written by Mark Drela, to design the blade sections and estimate performance.

Thesis Supervisor: Professor Jack Kerrebrock
Title: Richard Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank Professor Jack Kerrebrock, for guiding me through my work here. I would also like to thank Mark Drela for writing the MISES code that made this type of design work possible. I would also like to thank the other professors that have also given me insight in my graduate work: Eugene Covert, Ed Greitzer, Alan Epstein, and Jaime Peraire. I cannot give enough thanks to Duncan Reijnen, my partner and mentor, whom I am leaving behind to see if this stuff really works. I would like to thank Don Cho, Pam Barry, Yong-Cheng Liang, Don Hoying, Dave Carr, Dave Underwood, and other students around the GTL who have listened to me ramble on about some idea or another. I would like to thank my family which has given me all the support I have needed, and I would like to thank my fiancee Jennifer Peskin, who has done so much for me, I cannot begin to describe it.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern high bypass ratio turbofan engines are made to give a certain level of thrust while meeting noise standards and using a minimum amount of fuel. To minimize fuel use, engines are designed with large bypass ratios, to take as much power as possible out of the core flow and put it into the bypass stream. The highest engine specific impulse comes when the amount of power taken from the core and put into the bypass makes the exit velocities of the bypass flow and core flow equal. Current commercial engine designs do not do this because noise requirements limit them to one stage in the fan. Using current technology, the pressure rise from a one stage fan cannot give the fan flow the optimum velocity. Using the technology of boundary layer suction, a design for a single stage fan will be proposed that gives a high enough pressure ratio to optimize the propulsive efficiency. This argument is developed in this chapter.

Chapter two describes the streamline curvature analysis which was used to compute the throughflow of the fan.

Chapter three describes the use of MISES to design the blade sections.

Chapter four describes the results of the design, compares it to a current technology fan, and gives recommendations for further work.

1.1 Motivation for high pressure ratio

One goal in designing a turbofan engine for a specific use is fuel efficiency. This goal can be met by increasing two types of efficiencies: thermal cycle efficiency and propulsive efficiency. For the turbofan/turbojet cycle, the thermal efficiency, η_t, is given by:

$$\eta_t = 1 - \frac{T_0}{T_3} \tag{1.1}$$

T_0 refers to the ambient static temperature, and T_3 refers to the static temperature at the compressor exit. See figure 1-1 for a schematic of the engine locations used in this chapter.

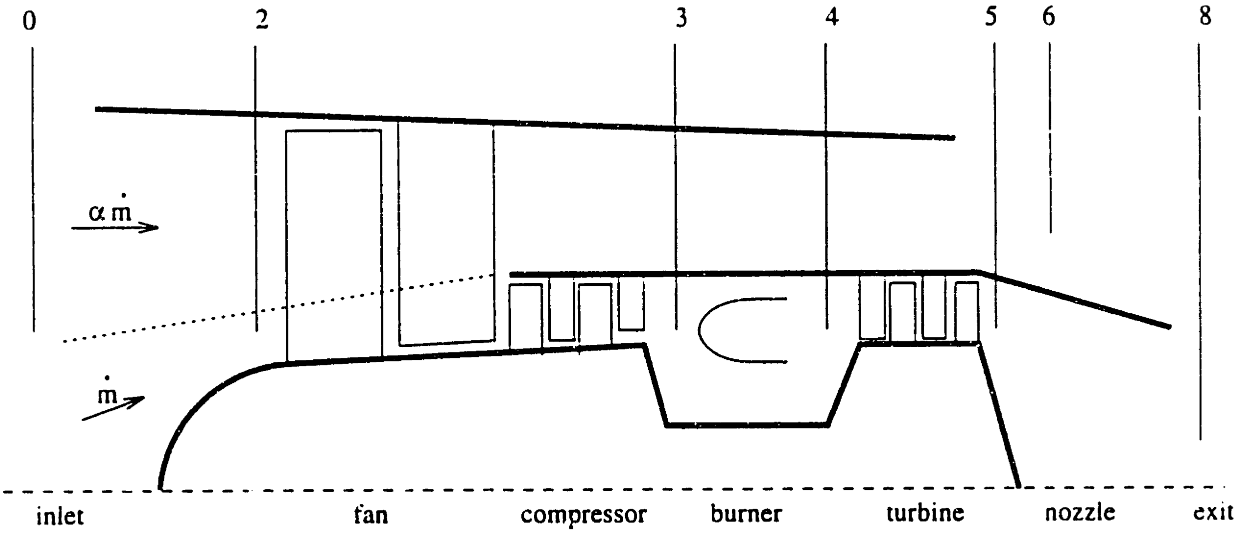

Figure 1-1: Schematic of a turbofan engine

This thermal efficiency depends only on the compressor temperature ratio, which is really a function of the overall compressor pressure ratio, π_c. As the total pressure ratio of the compressor increases, the efficiency of the engine rises until the temperature rise of the compressor is so great that it allows no energy to be added in the burner. This pressure ratio is generally fixed for a given technology, and may also be set as a function of turbine inlet temperature ratio, Θ_t, and Mach number to give the maximum thrust per unit airflow.

Propulsive efficiency, η_p, is defined as the ratio of the power delivered to the vehicle to the net power delivered to the engine flow, given by:

$$\eta_p = \frac{2u_0}{u_e + u_0} \tag{1.2}$$

where u_0 is the flight velocity and u_e is the mass averaged exit velocity from the engine.

As the engine mass averaged exit velocity decreases, the propulsive efficiency increases for a given flight velocity. In a turbofan engine, as more energy is taken from the core and put into the bypass stream the overall exit velocity decreases, so the propulsive efficiency goes up. The propulsive efficiency is maximized when the core and bypass streams have equal exit velocities. Energy can be taken from the core in two ways: increasing the bypass ratio, α, or increasing the pressure ratio of the bypass fan. Since commercial engines are limited to a single stage fan, the pressure ratio available from the fan is critical for efficiency.

The use proposed here for boundary layer suction would be to increase the pressure ratio of the bypass fan, thus increasing propulsive efficiency. Another advantage of boundary layer removal from the compressor would be that high entropy air takes more work to compress in the later stages of the compressor, so overall compressor efficiency can be increased by suction. A description of this effect can be found in [2], and will not be considered here. To find the optimum pressure ratio of the fan, we must make assumptions about the performance of the other components of the engine, set the bypass and core velocities equal, then solve for the fan pressure ratio. If we assume that the fan and core exit nozzles are choked, which is true at cruise, we get the result:

$$\tau_f = \frac{\frac{C_{pt}\Theta_t}{C_{pc}\Theta_0} + \left(\frac{1}{1+f}\right)(1 + \alpha - \tau_c)}{\frac{\left(\frac{\gamma+1}{\gamma-1}\right)_t}{\left(\frac{\gamma+1}{\gamma-1}\right)_c} + \alpha\left(\frac{1}{1+f}\right)} \tag{1.3}$$

f is the fuel to air mass ratio, which is equal to:

$$f = \frac{\overline{C}_p T_0}{\eta_b h}\left[(1+f)\,\Theta_t - \Theta_0 \tau_c\right] \tag{1.4}$$

It should be noted that α, the bypass ratio, is defined as the ratio of the bypass mass flow to the core mass flow. This value would change for an engine utilizing suction because of mass removal, but I have assumed here that the effects of the suction on the value of the bypass ratio is small.

In this case, fan thrust is given by:

$$\frac{F_8}{\dot{m}u_0} = \alpha\left[\frac{u_8}{u_0} - 1 + \frac{1}{\gamma_c M_0^2}\frac{T_8}{T_0}\frac{u_0}{u_8}\left(1 - \frac{p_0}{p_8}\right)\right] \tag{1.5}$$

where:

$$\frac{T_8}{T_0} = \frac{\Theta_0 \tau_f}{1 + \frac{\gamma_c - 1}{2}M_8^2}$$

$$1 + \frac{\gamma_c - 1}{2}M_8^2 = \left(\frac{p_0}{p_8}\delta_0 \pi_d \pi_f\right)^{\frac{\gamma_c - 1}{\gamma_c}}$$

$$\frac{u_8}{u_0} = \frac{M_8}{M_0}\sqrt{\frac{T_8}{T_0}}$$

The core thrust is given by:

$$\frac{F_6}{\dot{m}u_0} = (1+f)\frac{u_6}{u_0} - 1 + \frac{1+f}{\gamma_c M_0^2}\frac{R_t}{R_c}\frac{T_6}{T_0}\frac{u_0}{u_6}\left(1 - \frac{p_0}{p_6}\right) \tag{1.6}$$

where:

$$\frac{T_6}{T_0} = \frac{\Theta_t \tau_t}{1 + \frac{\gamma_t - 1}{2}M_6^2}$$

$$1 + \frac{\gamma_t - 1}{2}M_6^2 = \left(\frac{p_0}{p_6}\delta_0 \pi_d \pi_c \pi_b \pi_t\right)^{\frac{\gamma_t - 1}{\gamma_t}}$$

$$\frac{u_6}{u_0} = \frac{M_6}{M_0}\sqrt{\frac{\gamma_t R_t T_6}{\gamma_c R_c T_0}}$$

Total thrust can be computed by adding the fan and core thrusts, and the specific impulse is given by;

$$I = \frac{F}{g\dot{m}_f} = \frac{a_0(1+\alpha)}{g}\frac{F}{\dot{m}a_0(1+\alpha)}\frac{1}{f} \tag{1.7}$$

Values used here are typical of a next-generation, high-bypass ratio commercial

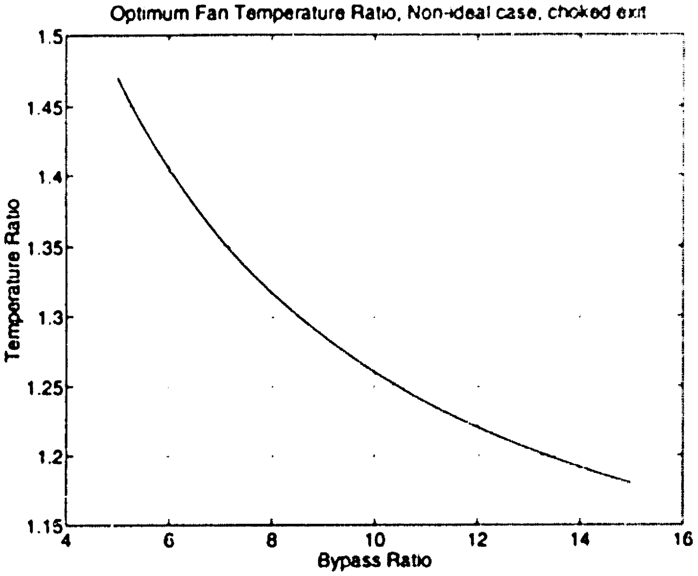

Figure 1-2: Optimum fan temperature ratio for a range of bypass ratios

turbofan; $M_0 = 0.8$, $T_0 = 222$ K, $g = 9.8$ m/s^2, $R = 287$ J/kg K, $\gamma_c = 1.4$, $\gamma_t = 1.34$, $C_{pc} = 1000$ J/kg K, $C_{pt} = 1130$ J/kg K, $h = 43{,}000{,}000$ J/kg, $\Theta_t = 7.5$, $\pi_c = 30.0$, $\pi_d = 0.95$, $\pi_b = 0.95$, $\eta_{poly} = 0.90$, $\eta_b = 0.95$, and $\eta_t = 0.9$.

Figure 1-2 shows the optimum fan temperature ratio for a range of bypass ratios. The optimum fan temperature ratio drops as the bypass ratio increases because more energy is taken from the core flow as the bypass ratio increases, so less work has to be done on the flow to equalize the flow velocities. A bypass ratio of ten was selected as typical for a next generation engine.

Figures 1-3 and 1-4 show the change in thrust per unit airflow and specific impulse as the bypass ratio changes. These are plotted for matched jet velocities. The thrust per unit airflow drops as bypass ratio increases because more air is being drawn through for the same amount of energy added. The specific impulse increases because more energy is being taken from the core as bypass ratio increases.

Figure 1-5 shows how the thrust per unit airflow changes as the temperature ratio of the fan is varied for a bypass ratio of ten. The optimum value comes when the jet velocities are equalized. At fan temperature ratios that are too high, the thrust drops off rapidly because the core starts losing thrust and eventually cannot provide

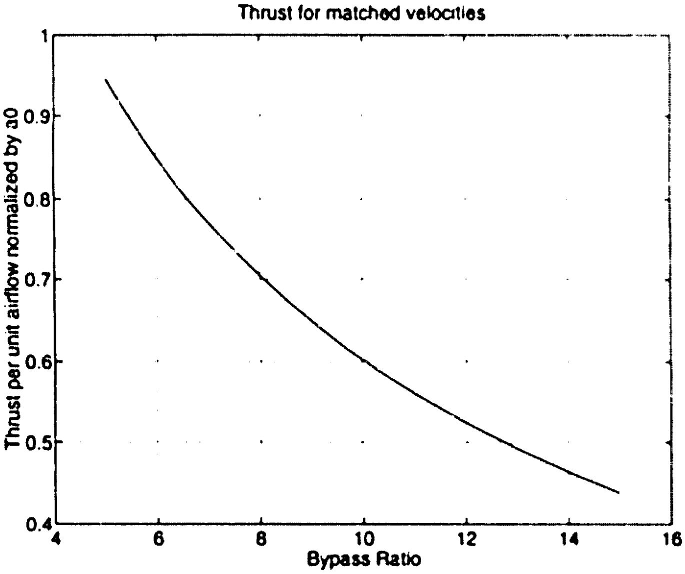

Figure 1-3: Thrust per unit of airflow for matched jet velocities

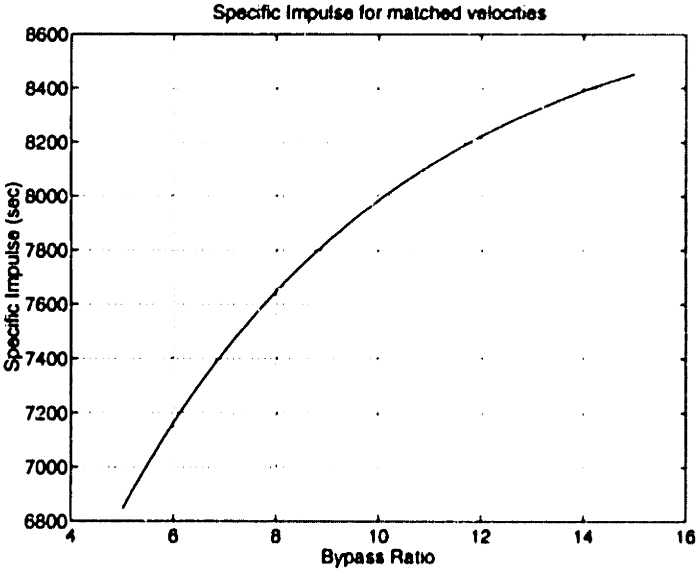

Figure 1-4: Specific Impulse for matched jet velocities

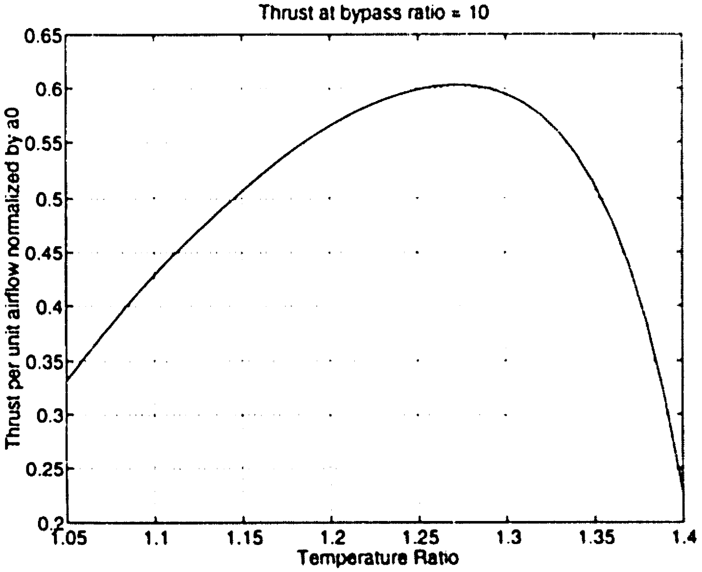

Figure 1-5: Thrust per unit of airflow for a range of temperature ratios

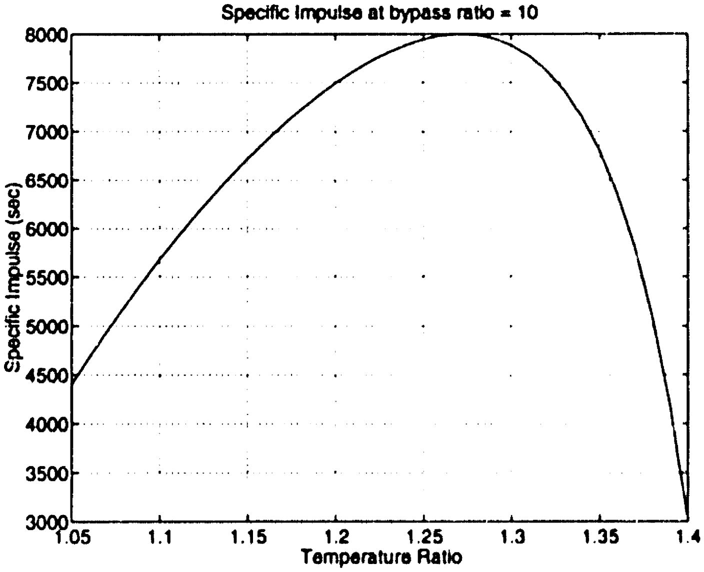

Figure 1-6: Specific Impulse for a range of temperature ratios

τ_f	Specific Impulse (s)	Specific Thrust
1.15	6722	0.507
1.23	7813	0.589
1.27	8002	0.603

Table 1.1: Comparison of high pressure fan with current fans

enough power to sustain the temperature ratio desired in the fan. Figure 1-6 shows how specific impulse changes with fan temperature ratio. This plot follows the same pattern as the thrust variation plot, and has its optimum at the same point.

For a bypass ratio of ten, the optimum temperature ratio is 1.27. Current fans provide temperature ratios around 1.15. By comparing the thrust and specific impulse of the fans, we can get an idea of what sort of advantage is gained by increasing the fan temperature ratio to optimum. Table 1.1 shows that a nineteen percent increase in specific thrust and specific impulse is possible. The temperature ratio for the fan designed here is 1.23. This value gives a sixteen percent increase in specific impulse and specific thrust. The slightly higher performance of a fan with $\tau_s = 1.27$ is not worth the difficulty of creating a fan to provide the necessary turning.

A disadvantage that would cancel out some of the gains possible in engine size and fuel consumption is the fact that with a higher fan pressure ratio, the turbine must be larger to provide enough power to turn the fan. The power required to turn the fan is given by:

$$\text{FanPower} = \alpha \dot{m} C_{pc} \left(T_{t7} - T_{t2} \right) \tag{1.8}$$

Dividing through by $\alpha \dot{m} C_{pc} T_0$ gives:

$$\text{FanPower} \propto \left(\tau_f - 1 \right) \tag{1.9}$$

This proportionality tells us that the low pressure turbine must increase in size to power a fan with a temperature ratio of 1.23 instead of one with a temperature ratio of 1.15. The power required by the fan increases by 53 percent. However, the advantages

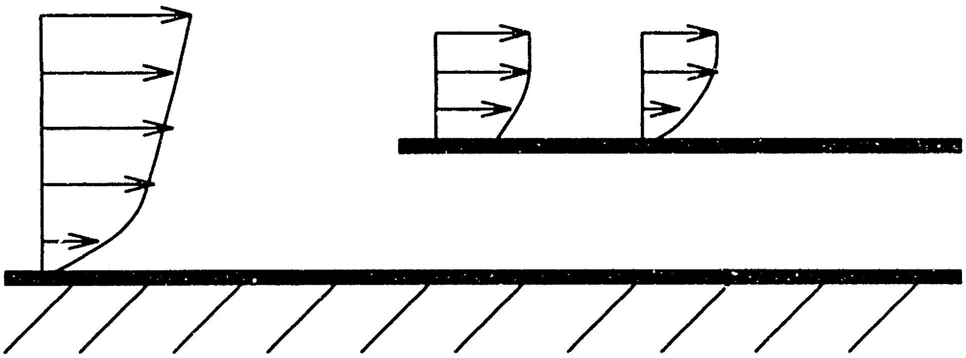

Figure 1-7: Boundary layer behavior at a scoop

of the increase in specific thrust will allow a smaller engine, which should more than make up for the larger turbine.

1.2 Usage of boundary layer suction

Previous experimental studies of boundary layer suction have shown that beneficial results can be obtained from suction in the correct places [8]. The suction would be applied at the point along the suction surface of the blade where the boundary layer is near separation. It is likely that this would also be near the point where the passage shock hits the surface of the blade. The pressure rise across the shock would thicken the boundary layer quickly. A possible advantage that has not been considered here is that the placement of suction could stabilize the shock position, reducing unsteadiness and noise in the compressor.

The suction would take the form of a scoop (see figure 1-7). This type of suction would provide the best means of restarting the boundary layer, since it would almost guarantee that none of the air that is sucked off would reenter the flow. Use of a porous surface or suction holes would create a mixing region, and would not suck off the boundary layer as cleanly as a scoop.

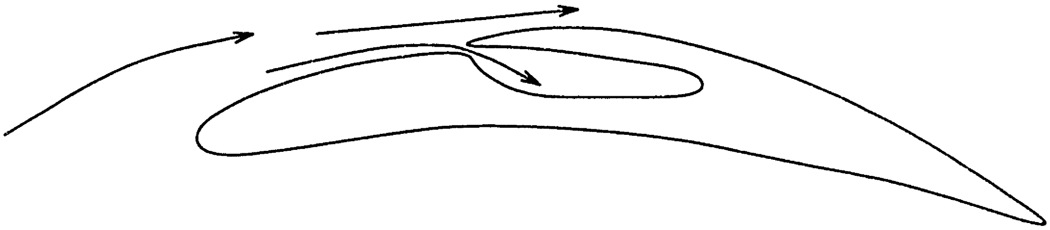

Figure 1-8: Blade cross section with a boundary layer scoop

1.3 Design procedure

The procedure used to design the fan consisted of three parts. The first part was to choose an engine type, make some assumptions about its performance, and then compute the fan temperature ratio desired for optimum efficiency. This was done in section 1.1. The second step in the design was to compute the streamline locations for a rotor and stator that would perform as computed in the first step. This was done with a streamline curvature computation, described in chapter 2. These streamline locations would be used in the third step of the design, described in chapter 3, which is to use MISES in the quasi-3D design mode to design the blade cross-sections that go along the streamline paths previously computed. Once the flow along all the streamlines has been computed, loss factors and other performance metrics can be computed. If the design becomes unworkable at any stage of the design process, iteration at an earlier step would be used to modify the design. A full 3D code would be used only to validate the findings of MISES, and would not be necessary as an iteration in the design process because the 3D streamtube interaction has been accounted for by the streamline curvature code and MISES.

The fan design is intended as one that could be used on commercial jet engines, so there are some limitations on the fan parameters that went into the design. Since the noise has to be kept low, the rotational speed of the fan is Mach 1 at the tip. It would have been better to make the fan even slower so that the relative Mach number of the incoming air was below Mach 1, but this was not feasible. The fan is also designed to give constant work across its span. This is done to keep the design simple. A varying temperature ratio may be advantageous if, for example, one cannot get the higher temperature ratio at the hub of the fan, so that the higher temperature ratio is only used in the bypass flow. Although that could increase the bypass velocity towards optimum, mixing after the rotor would degrade the effectiveness of such an approach.

Some limits to the design were imposed in the streamline curvature program. These could be relaxed with some modifications to the SC code. Outside of the rotor and stator areas, there is no swirl in the flow. This corresponds to having no inlet guide vanes, and having the stator return the flow to axial. Both of these conditions are desirable in the fan stage of an engine. The lack of inlet guide vanes increases the flow per unit area and reduces noise, and the return to axial flow is used because any swirl velocity in the bypass flow exit will be energy wasted. The flow quantities that change through the rotor and stator like enthalpy and entropy were assumed to change linearly through the rotor and stator.

The position of the scoop and the flow along each streamsurface is found by use of the MISES solver. This code, as modified by Duncan Reijnen, can predict the effects of suction on a stream surface that is changing position in a rotating compressor. The boundary layer solution in MISES can be used to predict separation and loss generation. Suction is not modeled in the streamline curvature code because the small amount removed should not have an effect on the streamline curvature. The modifications to MISES to model suction will be described in chapter 3.

Chapter 2

Streamline Curvature Analysis

2.1 Purpose of analysis

After the fan pressure ratio and size were decided upon, the next step was to compute the streamlines that go through the fan. The streamline curvature analysis would give an estimate of the turning needed from the streams, the Mach number of the flow, the diffusion factor on each blade, and the contraction desired from the fan's duct. Another important piece of information given by the code is the actual locations of the streamlines. The quasi-3D analysis done by MISES assumes the flow moves along a stream tube that is changing its radial position and may be contracting or expanding. The streamline curvature analysis computed the radial location and width of the streamtubes that go through the rotor.

2.2 Structure of code

The streamline curvature analysis is done in the r-m coordinate system. The r coordinate refers to the distance from the hub, and the m coordinate refers to the distance along a streamline. This coordinate system is illustrated in figure 2-1.

The streamline curvature equation (2.1) tells us the change in streamwise velocity across the compressor annulus in this coordinate system. A derivation of this equation can be found in [7].

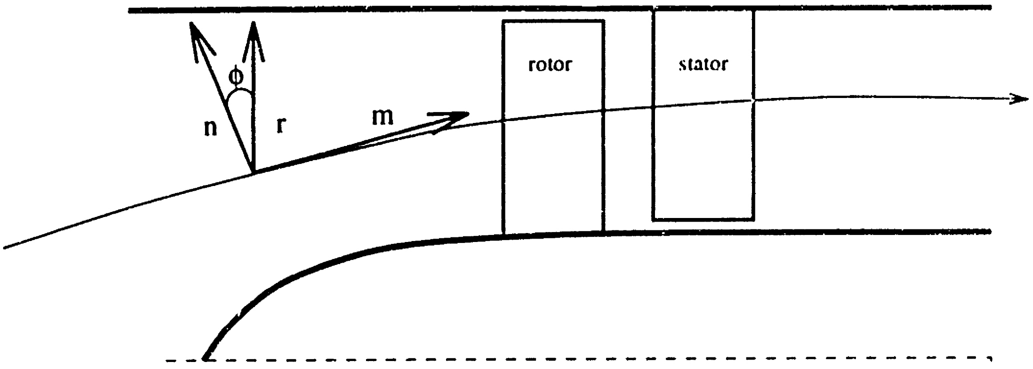

Figure 2-1: r-m coordinate system for streamline curvature analysis

$$\frac{1}{2}\frac{\partial}{\partial r}\left(v_m^2\right) = \frac{\partial h_t}{\partial r} - T\frac{\partial s}{\partial r} + v_m\frac{\partial v_m}{\partial m}\sin\phi + \frac{v_m^2}{r_c}\cos\phi - \frac{1}{2r^2}\frac{\partial\left(r^2 v_\theta^2\right)}{\partial r} + \frac{v_m}{r}\frac{\partial}{\partial m}\left(r v_m\right)\tan\epsilon \tag{2.1}$$

The term $\frac{1}{2}\frac{\partial}{\partial r}\left(v_m^2\right)$ gives the change in velocity across the annulus. The $\frac{\partial h_t}{\partial r}$ term refers to the change in enthalpy across the radius, which is zero for a constant work fan. The $T\frac{\partial s}{\partial r}$ term refers to the change in entropy across the annulus, which could be due to differences in loss from hub to tip. The term $v_m\frac{\partial v_m}{\partial m}\sin\phi$ refers to the component of the streamwise acceleration in the r direction. The $\frac{v_m^2}{r_c}\cos\phi$ term is due to the pressure gradient from streamline curvature. The $\frac{1}{2r^2}\frac{\partial\left(r^2 v_\theta^2\right)}{\partial r}$ term is from the change in angular momentum across the annulus, which is zero for a constant work (free vortex) fan. The term $\frac{v_m}{r}\frac{\partial}{\partial m}\left(r v_m\right)\tan\epsilon$ refers to the mean radial pressure gradient created when the blades are angled in the tangential direction with angle ϵ. Here it is assumed that the blades have no tangential lean, so this term was dropped from the code. This is not exactly true as the blades will have some local lean due to the change in blade cross section from hub to tip. The SC code does compute the two terms that should vanish for a constant work fan.

To use this equation to compute the change in streamwise velocity across the radius, h_t, v_θ, and s must be specified throughout the flow field. h_t is specified so that it changes across the rotor linearly and is constant everywhere else. s changes through the rotor and stator by the inclusion of a loss factor, $\overline{\omega}$, which is used to compute the entropy change [7]. v_θ is computed from the local enthalpy and the Euler turbine equation 2.2.

$$c_p\left(T_{tc}-T_{tb}\right)=\omega\left(r_c v_c-r_b v_b\right) \tag{2.2}$$

With equations 2.1, 2.2, and the flow definitions the code computes the change in velocity across the annulus. However, this does not satisfy conservation of mass. We must apply equation 2.3 explicitly across the annulus to ensure that mass is conserved.

$$2\pi\int_{r_H}^{r_T}\mathrm{Bl}\left(r\right)\rho v_m\cos\phi\, r dr=\dot{m} \tag{2.3}$$

To solve for the flow through the duct, the code starts at the iniet and marches downstream. At each meridional station, the code uses equation 2.1 to compute the change in v_m across the annulus. These velocities are then scaled to conserve mass according to equation 2.3. Then the streamlines are displaced so that the mass flowing through each streamtube is constant. Then the code returns to the beginning of this procedure until convergence is achieved at the meridional station. After the end of the duct is reached, the code iterates down the duct again until the streamlines converge on a radial location.

Source code for the SC program can be found in appendix A, and details of the algorithm and solution procedures can be found in references [7], [10], [6], [9] and [5].

2.3 Code results

The final design selected for the design has a hub to tip ratio of 0.55. The final computation grid for the fan passage is shown in figure 2-2. The fan duct was designed to keep the axial Mach number approximately constant, and was finalized by iterating back and forth between MISES. The major parameters that had to be tested in MISES

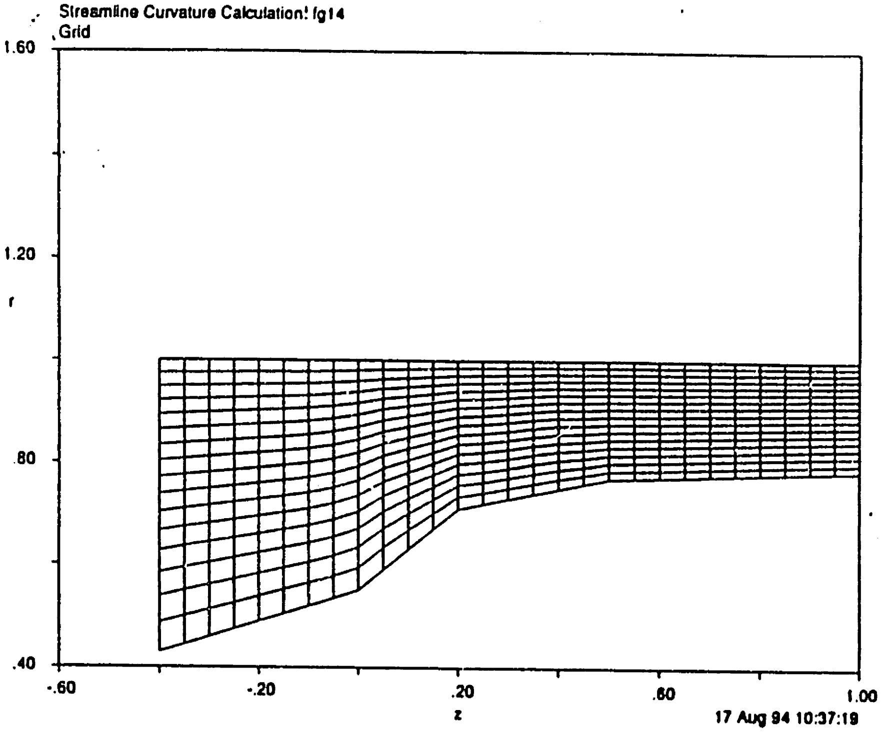

Figure 2-2: Computation grid for the fan

were the numbers of rotors and stators, to change the solidity, and the overall hub to tip ratio, which would change the amount of turning necessary at the hub. The grid has 17 radial computation stations (streamlines) and 29 axial computation stations. The turning level chosen for the rotor combined with the hub to tip ratio keeps the rotor absolute frame exit velocity at the hub subsonic, as shown in figure 2-3.

The diffusion factor for a blade row measures the loading on a blade, which can be correlated to losses. The fan diffusion factors computed by SC range from 0.56 to 0.69, as shown in figure 2-4. This level of diffusion would imply unacceptable losses for a fan that did not use boundary layer control, but since the diffusion factor is related to boundary layer growth, a scoop that restarts the boundary layer makes such a high diffusion factor acceptable. The design has 32 rotor blades and 49 stator blades around the annulus.

Another important parameter computed by SC across the rotor and stator is the axial velocity-density ratio (AVDR). The AVDR is simply the ratio of the streamtube area at the fan inlet to that at the fan exit. The greater the AVDR, the more

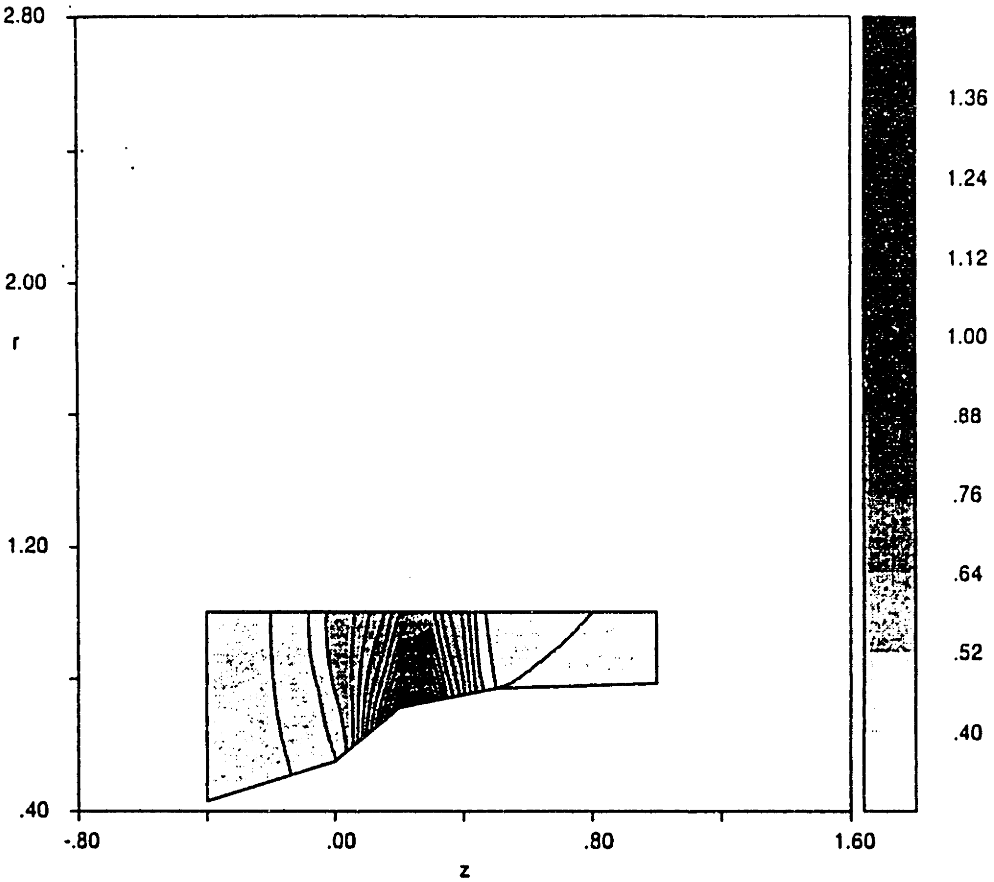

Figure 2-3: Duct total Mach number

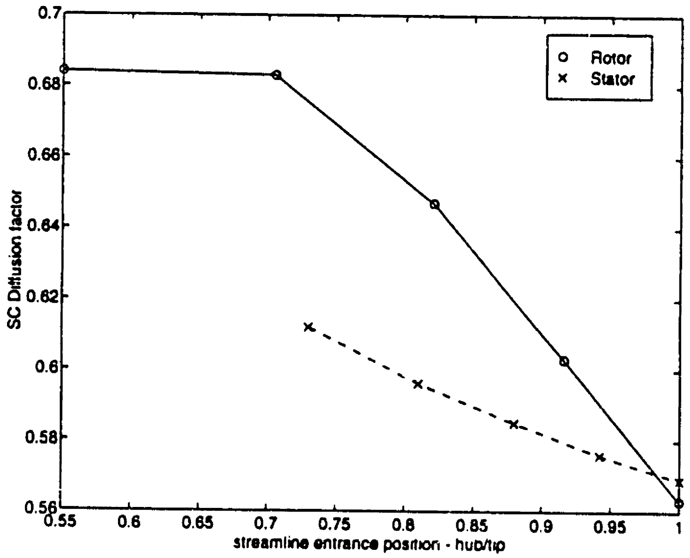

Figure 2-4: Diffusion factor across the fan

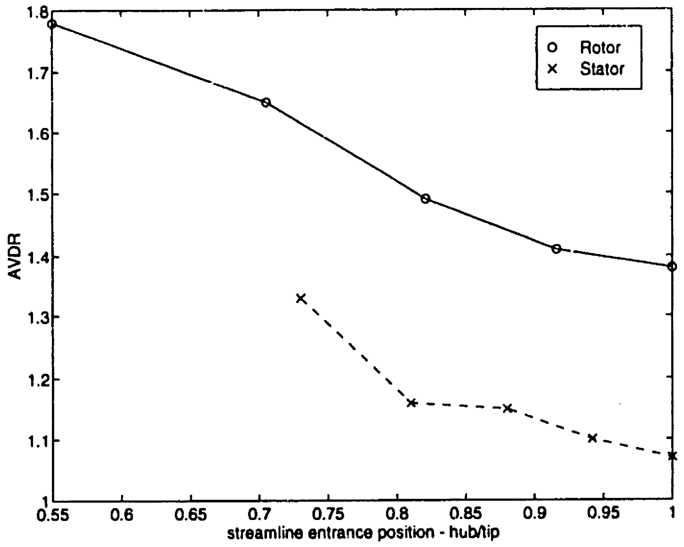

Figure 2-5: AVDR across the fan

streamtube contraction there is, and the more risk of choking. This fan has a greater AVDR than normal because in general, fans are designed to keep the axial velocity constant, and with a higher pressure ratio than most fans, the streamtubes must contract more than average. The AVDR across the fan is shown in figure 2-5. The duct streamwise Mach number is shown in figure 2-6.

SC also computes the velocity triangles for the streamlines in the fan. Five streamlines were chosen to be computed in MISES. 1, 5, 9, 13, and 17. These correspond to the hub, quarter-span, mid-span, three-quarter-span, and the tip streamlines. One-fourth of the total mass flow passes between adjacent computed streamlines. The velocity triangles for the rotor and stator along those five streamlines can be found in appendix B.

2.4 Blade section generation

The SC code contains a procedure that generates blade sections for use with the MISES analysis program. The blade sections generated are the best guess that SC can make to approximate the performance necessary for the fan to perform the amount of

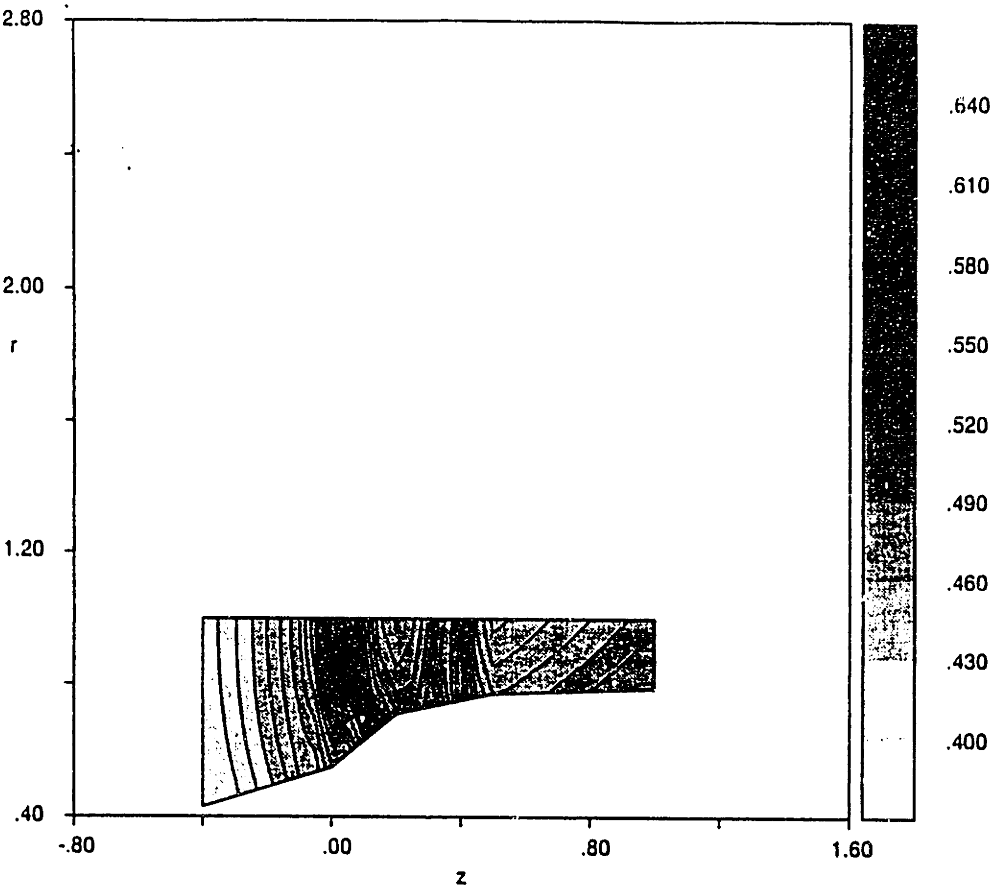

Figure 2-6: Duct streamwise Mach number

work that is required of the rotor at an acceptable level of loss. For MISES to analyze a blade section in quasi-3D mode, it needs three files: BLADE.xxx, STREAM.xxx, and ISES.xxx, where xxx is the file suffix which identifies the blade section. BLADE.xxx contains the blade cross-sectional geometry, initial inlet and exit flow angles, and the distance upstream and downstream of the blades to end the grid. STREAM.xxx contains the streamtube thicknesses, positions, and rotational speed for quasi-3D analysis. ISES.xxx contains the global variables and constraints as well as other numerical parameters that MISES uses.

The blade cross section is generated as an estimate of what blade shape can produce the required turning levels under the conditions computed by SC. The blade shape used is known as a multiple circular arc (MCA). The blade is defined by two arcs that make up the upper surface and a single circular arc to make the lower surface. The nose of the blade is a half circle, inclined at the flow entrance angle (β_1') with a zero to negative two degree angle of incidence. The critical consideration when giving an initial guess at the blade shape was to prevent choking. If the blade

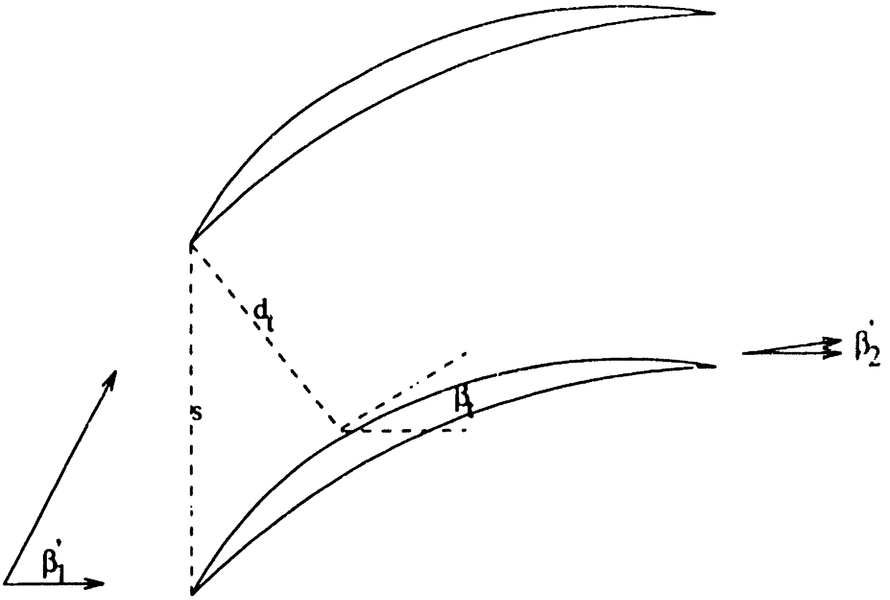

Figure 2-7: Blade passage showing throat and flow angles

chokes, no initial solution can be obtained by MISES, so the blade cannot even be redesigned to allow more mass flow. SC does not check if a blade passage chokes. Such a calculation is possible, but because SC is a design program, it is quicker to simply generate blade sections that will not choke, instead of checking for choked conditions with specified blades.

The critical point where the blade tends to choke (the throat) is where the normal line across the passage touches the leading edge of one blade and the suction surface of the blade next to it. The critical design objective is to make this throat width large enough to pass the incoming mass flow. An analysis using a throat area computation method shown by Davis and Millar in [1] is done when generating the blades to ensure that they will not choke. Although the width in the circumferential direction is set by the spacing, the flow can be turned in the entry region so that the flow angle is closer to axial and the flow has more normal area to pass through. The critical flow angle at the throat, β_{max}, is computed as follows:

$$x_t = s\sin(\beta_1')$$

$$\frac{A}{A^*} = \frac{1}{M_1'} \left(\frac{1 + \frac{\gamma-1}{2} (M_1')^2}{\frac{\gamma+1}{2}} \right)^{\frac{\gamma+1}{2(\gamma-1)}}$$

Streamtube height at the throat, h_t, is computed by linearly interpolating between inlet streamtube height, h_1, and exit streamtube height, h_2.

$$d_1 = s \cos(\beta_1')$$

$$A_1 = d_1 h_1$$

A_{min} is the minimum area possible that will pass the required mass flow. β_{max} is the largest flow angle allowable at the throat that will give at least A_{min} for the flow to pass through.

$$A_{min} = \frac{A_1}{\frac{A}{A^*}}$$

$$\beta_{max} = \arccos\left(\frac{A_{min}}{h_t s}\right)$$

$$\beta_t = \beta_{max} - \frac{\beta_1' - \beta_{ex}}{8}$$

β_{ex} is the exit slope of the flow plus a deviation angle which ranged from twenty-five to forty percent of the desired turning angle, and is computed as follows for thirty percent deviation:

$$\beta_{ex} = \beta_2' - 0.30\,(\beta_1' - \beta_2')$$

$$\Delta = \frac{s}{2}\sin(\beta_1' + \beta_t)$$

The beginning slope β_1', and ending slope β_t, along with the axial distance Δ, gives enough information to define the arc of a circle. If the circle's center is assumed to be at (0,0), and the two points of the arc are (x_0,y_0) and (x_1,y_1), then:

$$x_0 = -\Delta\frac{\sin(\beta_1')}{\sin(\beta_1') - \sin(\beta_t)}$$

$$x_1 = x_0 + \Delta$$

$$y_0 = \frac{-x_0}{tan(\beta_1')}$$

$$y_1 = \frac{-x_1}{tan(\beta_t)}$$

The rear section of the suction surface is defined similarly, with the arc going from β_t to β_{ex} and the axial distance of the arc given as the remainder of the meridional chord distance.

The lower (pressure) side of the blade is given as a single arc. This arc has the beginning and ending points defined exactly, because they have to match the upper surface endpoints. The inlet slope of this arc is defined to be equal to the slope of the inlet flow, β_1', minus a constant number of degrees. This gives a larger wedge angle to the underside of the nose, placing some compression on the flow after it passes through the throat. In the rotor, no wedge angle is added. In the stator, the wedge angle went from zero to five degrees, depending on the case. With this information, the center of the circle is found to be at:

$$x_c = \frac{x_1^2 - x_0^2 + (y_1 - y_0)^2 - \frac{2(y_1 - y_0)x_0}{\tan\left(\beta_1'\right)}}{2\left(x_1 - x_0\right) - \frac{2(y_1 - y_0)}{\tan\left(\beta_1'\right)}}$$

$$y_c = y_0 - \frac{x_c - x_0}{\tan\left(\beta_1'\right)}$$

The coordinates generated by these arcs are used to make the BLADE.xxx file for MISES. The STREAM.xxx file is made from the streamtube thicknesses and locations. The R coordinate is given by the y location of the streamline normalized by the chord. The value of B, the streamtube thickness is given by taking the difference between the y position of the two streams above and below the desired streamline. For the hub and tip streamlines, the streamtube thickness is given by the difference in y of the stream itself and the next stream towards the interior. Before and after the blade passage, the streamtube thickness is given to MISES as constant. This is done because if MISES were given the actual computed stream thickness up and downstream, the flow would accelerate as it approaches the blade (for subsonic relative Mach numbers) because of the streamtube contraction, and the mass flow through the blade would be greater than what it ought to be, thus the inlet plane Mach number would have to be adjusted for this effect. To get an accurate model of what the flow is like in the blade passages, MISES is given no stream tube contraction before or after the blade passage.

The ISES.xxx file, containing the Mach numbers, flow angles, Reynolds number, boundary conditions, and other parameters dealing with the MISES numerics, is also generated from information computed by SC. The global constraints and global variables chosen to be used in MISES were chosen so that the mass flow and entry angle could vary at the grid edge, but the characteristic is held constant, so there is no actual work being done before the grid inlet. For these computations, global constraints 16, 3, 4, and 18 are used. These correspond to the Kutta conditions (continuous pressure) at leading and trailing edges, a fixed inlet flow Mach number, and the exit static pressure being fixed. The MISES global variables are 1, 2, 5, and 15, which correspond to allowing inlet angle, exit angle, total inlet mass flow, and the location of the leading edge stagnation point to vary.

Chapter 3

Blade Section Design

3.1 MISES design code

MISES, Multiple Interacting Streamtube Euler Solver, is a coupled viscid/inviscid flow solver that can operate in either design or analysis mode. The inviscid flow is solved using the steady Euler equations, and the viscous portion of the flow is solved using integral boundary layer equations that march downstream. The Newton-Raphson linearization technique is used to solve the inviscid flow equations. The results from the inviscid flow are used to compute a boundary layer. The inviscid flow is then displaced by the boundary layer displacement thickness, δ^*, and the program will iterate in this fashion until a solution is converged upon. The three dimensional effects of streamtube contraction and rotation are included in the MISES calculations.

In analysis mode, the code will take a blade of a given geometry and boundary conditions, and compute the Mach and pressure distribution in the flow, as well as loss and shock information. In the design (inverse) mode, the code will take a given surface Mach number distribution and modify the blade geometry to minimize the error from that distribution. The code will also operate in a mixed mode, where part of the blade has the geometry specified, and the rest of the blade has the Mach number specified.

Details of how MISES works can be found in previous works [3] [11].

The modification to the code that was made for this work was an addition of suction effects, done by Duncan Reijnen. Suction on a blade would have two effects: delay of boundary layer separation and mass removal. The delay in separation has been modeled through a reduction in the momentum thickness, θ over a few grid points in the domain. MISES applies three equations to compute the boundary layer: the Von Karman integral momentum equation, a shape factor equation derived from the integral kinetic energy equation, and a dissipation lag equation in turbulent regions. In laminar regions, a transition equation replaces the dissipation lag equation. These equations and derivations of them can be found an appendix B of Youngren's report [11]. These equations are solved by logs, and if their residual is driven to a factor instead of to zero, this simulates a reduction of the boundary layer momentum thickness, θ. The reduction of θ reduces the boundary layer thickness and shape factor, defined as $H = \delta^*/\theta$. A reduced shape factor is indicative of a boundary layer that has a fuller profile and is less likely to separate. The mass removal is modeled by subtracting the height of the scoop from the height of the blade. This can result in a negative blade thickness at the blade trailing edge and grid overlap in the wake zone behind the blade. The mass flow that is in the overlapping zone is considered to have been removed.

3.2 Rotor and Stator blade section choice

The initial blade section choice was to use a double circular arc (DCA) blade. A DCA blade is defined by a circular arc that makes up each of the bottom and top surfaces. The arcs were created by drawing a circular arc with the inlet flow angle as the incoming angle and the exit flow angle plus a deviation as the exit arc angle. This arc gave the beginning and ending points of the blade, and then by assuming a midspan thickness, a third point was placed on the top and bottom surfaces. These points would define the arc for each of the top and bottom surfaces. This design ended up being generally unanalyzable. The blades would choke, and MISES would be unable to converge on a solution. This happened because the incoming flow would be deflected upward, away from the axial direction, thus reducing the effective flow area. The streamtube contraction also reduced the flow area, and since the Mach numbers were generally near one, the flow choked easily. The next attempt was to eliminate the flow compression on the upper surface in the entry region of the blade before the throat. This was done by making that entry surface straight, then making a circular arc for the rear portion of the upper surface, and another arc for the lower surface. This blade still choked, because although the flow width was constant, causing no contraction, the streamtube contraction in the spanwise direction was still too great for the flow to tolerate without choking. The design used for this fan has a multiple circular arc (MCA) geometry, as described in section 2.4. The flow in the inlet region turns toward the axial direction so that the flow does not choke.

3.3 Rotor and Stator blade section design process

Once the necessary files are created by SC, computation grids must be generated for the blade sections. The grid type used was the standard grid, as opposed to the other grid option in the MISES grid generator, known as the offset-periodic grid. The offset periodic grid has separate blocks for each part of the blade, so the normal grid lines are more perpendicular to the flow direction. This makes the grid blocks more rectangular, so the shock resolution is better and the computation is smoother around the leading edge of the blade. The drawback of the offset-periodic grid type is that it takes approximately 5 times longer to solve a case than on the standard grid because the matrix that is made by MISES has a larger bandwidth (more nonzero diagonals). The standard grid was used, and it seems that the results were satisfactory. Some extra points were clustered around the nose by changing the curvature weighting exponent in the grid generator to 0.8.

These sections were then analyzed in MISES. The first step was to compute the solution with the given blade and no viscosity. MISES solves the Euler equations for the flow, accounting for the 3D effects (rotation, streamtube contraction and displacement). The boundary layer displacement is zero for the inviscid analysis.

After the quasi-3D inviscid solution was found for the MCA blade, redesign would

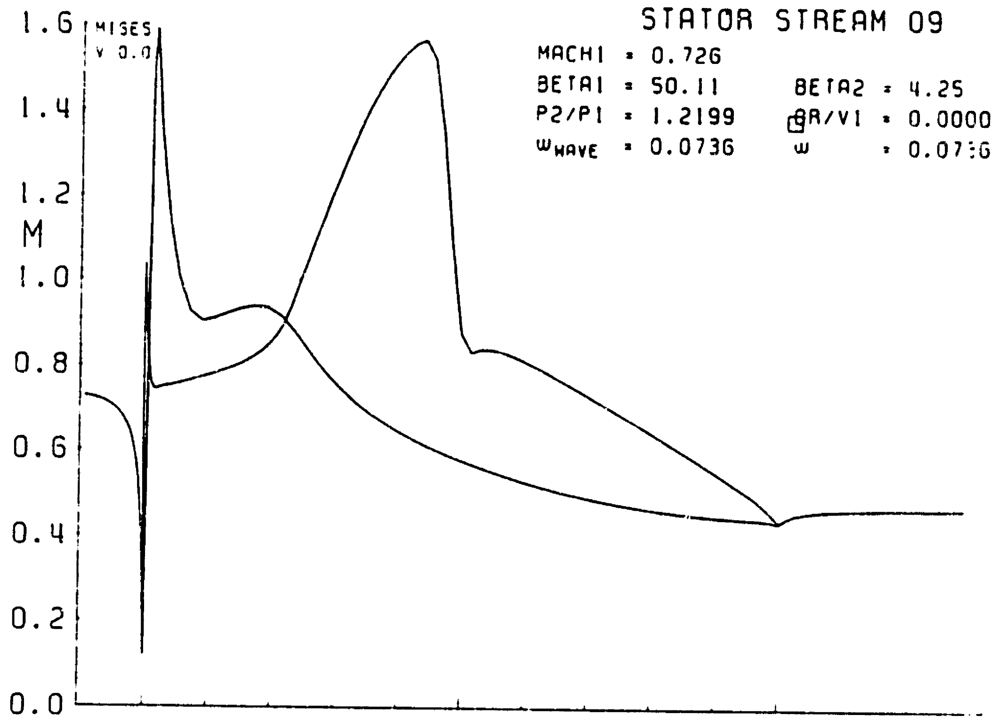

Figure 3-1: Stator blade before redesign

be done if the blade Mach profile seemed poor. For example, if the stator had a strong shock, it would be redesigned to make the shock weaker. An example of a blade section needing redesign is shown in figure 3-1. In that case, the shock that had formed was removed by a camber redesign. The code eliminated the acceleration that led to the shock by flattening the camber line. To redesign the blades, the Mach profile was modified to be smoother, then MISES was ran in mixed-inverse design mode, with global variables and constraints 11, 12, 13, and 14 selected. When iterating in design mode, the blade geometry would be changed to make the computed Mach profile match the input Mach profile.

After that redesign, a viscous solution would be computed. To make the solution easier, suction was added in a position estimated to do the most good. If there was a strong shock in the flow, suction would be near the shock position. If there was no shock, the suction would be placed at approximately 70-80 percent chord to allow the boundary layer to restart after the suction and prevent separation as the flow goes toward the trailing edge. The reduced wake thickness greatly decreases the computed loss. The placement and strength of the suction would be modified if necessary.

The last step in the design is to make sure the blade turns the flow to the correct angle, which would be the angle output by SC. The turning of the blade was driven to the correct value by choosing constraint 2, output flow angle, and variable 27, a pattern of geometry variation allowing the trailing edge to move. MISES would modify the geometry of the trailing edge to make the overall turning match the input value. The total temperature ratio (work) of the blade is then guaranteed to be correct because the input and output flow match the values given by SC. This step had to be applied one iteration at a time, because the calculation seemed to be unstable in this mode. After applying one iteration allowing the trailing edge to move, the geometry would be frozen, and the solution reconverged. This allowed the blade to be modified so that the flow exit angle would be within 2 degrees of the desired angle.

The streamline mach plots and suction side boundary layer thicknesses are given in appendix C.

3.4 Scoop Height Computation

The results presented here do not include the mass removal effects that were described in section 3.1. This is due to the fact that the solver did not behave very well when the scoop height was added into the computation scheme. The procedure attempted was to complete a design as described previously, then to write out the modified blade geometry file, compute the scoop height, generate a new grid, then recompute the solution. Unfortunately, the solution would generally not reconverge. One possible problem is the technique used to model the scoop. In the grid generator, the user is asked for the scoop height, but not where the scoop is placed. The program simply thins the blade linearly along the span until the full scoop height is reached at the blade trailing edge. When running the solver, the user is again asked for the scoop height, but this time, the program knows where the scoop is. A better approach would have the user specify only the percentage of δ^* or θ to remove, then have the code compute the proper scoop height and displace the inviscid flow by an additional amount to model the scoop mass removal.

The method used to compute the proper scoop height depends on a given profile, known as Cole's profile, which is that used by Drela in his code [4]. The profile has an assumed slip velocity at the wall, and increases to the freestream velocity at the edge of the boundary layer as the sine function. All of the equations that follow are in the MISES code. These were used to compute u_s, the wall slip velocity normalized by the edge velocity, as a function of nondimensional displacement thickness, nondimensional momentum thickness, Reynolds number, and boundary layer edge Mach number.

$$H = \frac{\delta^*}{\theta} \tag{3.1}$$

$$H_k = \frac{H - 0.29M_e^2}{1 + 0.113M_e^2} \tag{3.2}$$

$$Re_\theta = \theta Re_c \tag{3.3}$$

$$H_o = 3.0 + \frac{400}{Re_\theta} \tag{3.4}$$

$$H_r = \frac{H_o - H_k}{H_o - 1.0} \tag{3.5}$$

$$H_s = 0.5H_r^2 \frac{1.5}{H_k + 0.5} + 1.5 \tag{3.6}$$

$$u_s = 0.15H_s \left(3.0 - 4.0\frac{H_k - 1.0}{H}\right) \tag{3.7}$$

The following relations allow δ, θ, and the velocity throughout the boundary layer to be computed.

$$\delta = \theta \left(3.15 + \frac{1.72}{H_k - 1.0}\right) + \delta^* \tag{3.8}$$

$$\theta = \int_0^\infty \left(1 - \frac{u}{u_c}\right) \frac{u}{u_e} dx \tag{3.9}$$

$$\frac{u}{u_e}\left(\frac{x}{\delta}\right) = (1 - u_s)\sin\left(\frac{x}{\delta}\frac{\pi}{2}\right) + u_s \tag{3.10}$$

By substituting the velocity function into the integral, we get:

$$\theta = \int \left[\left(1 - u_s^2\right)\sin\left(\frac{x}{\delta}\frac{\pi}{2}\right) - \frac{(1-u_s)^2}{2}\cos\left(\frac{x}{\delta}\pi\right) - 0.5 + 2u_s - 0.5u_s^2\right] d\frac{x}{\delta} \tag{3.11}$$

If we integrate this from zero to one we get an expression for theta of this profile.

$$\theta = \left(1 - u_s^2\right)\frac{2}{\pi} - \left(0.5 - 2u_s + 0.5u_s^2\right) \tag{3.12}$$

The suction scoop is assumed to remove the lower portion of the boundary layer to reduce θ by a given amount. The suction leaves the top fraction of the boundary layer, with the height of the part remaining equal to $(1 - p_s)\,\theta$ where p_s is the percentage of θ that is removed. We integrate θ from $\frac{x}{\delta}$ to one and set that equal to the remainder of θ.

$$(1 - p_s)\,\theta = \left(1 - u_s^2\right)\frac{2}{\pi}\cos\left(\frac{x}{\delta}\frac{\pi}{2}\right) + \frac{(1-u_s)^2}{2\pi}\sin\left(\frac{x}{\delta}\pi\right) - \left(0.5 - 2u_s + 0.5u_s^2\right)\left(1 - \frac{x}{\delta}\right) \tag{3.13}$$

This can be solved iteratively for $\frac{x}{\delta}$ and that, multiplied by δ, gives the necessary scoop height to remove the desired portion of the momentum thickness.

The computed scoop heights and the suction locations are given in table 3.1. The mass percentage refers to the amount of the passage mass flow that is taken in by the scoop. The amount of mass sucked in by the scoop was estimated by taking the percentage of the passage width that was blocked by the scoop, and multiplying by the average velocity in the boundary layer, which is $\frac{u_s+1}{2}$. The mass averaged amount of mass removed in the rotor is 3.8 percent, and the average amount of mass removed

Streamline	Suction Amount (% θ)	Suction Position (%x/c)	Scoop Height (x/c)	Mass Percentage
Rotor hub†	-	-	-	-
Rotor 1/4 span	75	55	.0222	3.4
Rotor 1/2 span	75	40	.0279	3.2
Rotor 3/4 span	75	40	.0521	4.9
Rotor tip	50	40	.0386	3.4
Stator hub	50	70	.0379	8.7
Stator 1/4 span	50	75	.0379	7.2
Stator 1/2 span	50	80	.0501	8.3
Stator 3/4 span	75	80	.0507	8.2
Stator tip	85	80	.0549	8.5

† The rotor hub was not converged, so no suction was found.

Table 3.1: Suction percentage and scoop height

in the stator is 8.1 percent. This gives an overall stage mass removal of 11.6 percent of the inlet mass.

3.5 Performance Estimation

The loss is estimated by computing the loss along each streamline, then mass averaging. Although the total temperature ratio should be the same on each streamline, the losses and thus the total pressure ratio may differ as the hub section of the rotor has to do more turning of the fluid since it has a lower blade speed, for example. The loss factor on each streamline is the sum of the viscous and inviscid (shock) losses.

Stage efficiency, η_c is related to the loss factor by the following equation [7]:

$$\eta_c = 1 - \frac{\frac{\gamma-1}{\gamma}\left(\overline{\omega}'_b\left(1-\frac{P_b}{P'_{Tb}}\right)+\overline{\omega}_c\left(1-\frac{P_c}{P_{Tc}}\right)\right)}{\tau_s - 1} \tag{3.14}$$

$$\frac{P'_{Tb}}{P_b} = \left(1+\frac{\gamma-1}{2}M_b'^2\right)^{\frac{\gamma}{\gamma-1}}$$

$$\frac{P_{Tc}}{P_c} = \left(1 + \frac{\gamma - 1}{2} M_c^2\right)^{\frac{\gamma}{\gamma - 1}}$$

From equation 3.14, we can compute the stage efficiency along each streamline. Then the stage pressure ratio, π_c and polytropic efficiency, η_{poly} are computed from the following:

$$\eta_c = \frac{\pi_c^{\frac{\gamma-1}{\gamma}} - 1}{\tau_c - 1} \tag{3.15}$$

$$\eta_c = \frac{\pi_c^{\frac{\gamma-1}{\gamma}} - 1}{\pi_c^{\frac{\gamma-1}{\gamma \eta_p}} - 1} \tag{3.16}$$

The loss factors for each streamline are shown in table 3.2, and the efficiencies are shown in table 3.3.

Streamline	Loss Factor	Inlet Mach
Rotor hub	.0300 (est)	0.738
Rotor 1/4 span	.0199	0.889
Rotor 1/2 span	.0349	1.004
Rotor 3/4 span	.0474	1.095
Rotor tip	.0658	1.173
Rotor Average	.0375	
Stator hub	.0356	0.919
Stator 1/4 span	.0234	0.848
Stator 1/2 span	.0280	0.800
Stator 3/4 span	.0319	0.764
Stator tip	.0277	0.735
Stator Average	.0287	

Table 3.2: Loss factors

Streamline	τ_c	η_c	π_c	η_{poly}
Hub	1.23	0.953	1.995	0.953
1/4 span	1.23	0.969	2.017	0.969
1/2 span	1.23	0.948	1.995	0.953
3/4 span	1.23	0.934	1.976	0.940
Tip	1.23	0.919	1.957	0.926
Average	1.23	0.945	1.991	0.950

Table 3.3: Streamline efficiency and pressure ratio

Chapter 4

Summary and Conclusions

4.1 Engine system comparison

The baseline fan used for design comparison has a total temperature ratio of 1.15, a hub to tip ratio of 0.5, and a polytropic efficiency of 0.90. The fan designed here has a total temperature ratio of 1.23, and a hub to tip ratio of 0.55. The average computed polytropic efficiency is 0.95. Using the other engine parameters defined in section 1.1, the computed gain in specific impulse is 17.6 percent and the computed gain in thrust per unit of airflow is 22.9 percent. Even if the losses are actually higher than computed, and the average polytropic efficiency is only 0.90, the gain in specific impulse is 16.2 percent and the gain in thrust per unit airflow is 16.2. If the 7.0 percent reduction in fan area and the 11.5 percent mass removal are accounted for, the increase in thrust per unit diameter is 18.9 percent for the high efficiency case and 13.3 for the lower efficiency fan.

Although most of the performance gain comes from the increase in fan work, the reduction in loss also helps the overall system performance. The reason that a fan using suction can get lower loss with higher turning is that the viscous losses mostly show up in the wake, but the suction reduces the size of the wake, so the losses do not enter the flow.

4.2 Conclusions and Recommendations for further study

The design system used in this work is a very convenient and powerful mechanism for the design of turbomachinery. The combination of the streamline curvature code that can quickly generate a streamline pattern and velocity triangles given the amount of work and duct geometry with a fast quasi-3D solver with redesign capability allows an experienced user to get a preliminary design for a fan stage in less than a day. On the RS/6000 model 590 a streamline would take less than 2 minutes to converge on an initial solution, and each redesign or adding the boundary layer to the blade takes about the same amount of time.

The major problem with the design system is the lack of a dependable means for modeling the suction mass removal. A better process would be to integrate the scoop height calculation into MISES, and displace the inviscid flow by the correct amount. This would eliminate the need to find the boundary layer profile before the computation grid is generated. The means for inviscid flow displacement and boundary layer profile computation are already in the MISES code. It would also be desirable to integrate the suction position and strength parameters into the ISES.xxx file, instead of prompting the user for the information before each set of iterations.

The biggest fault in the design presented here is the unconverged solution at the hub of the rotor. It is possible that the high turning level at the hub causes an unsteady flow situation to exist, which MISES, a steady flow solver, cannot model. A more detailed 2D computational study would attempt to calculate the rotor performance at the hub.

Another useful task would be to validate the design system with a full 3D calculation of the rotor-stator flow. Ideally, the flow would be computed with both viscous effects and suction included. Modeling viscosity or suction alone would not be productive, because if only viscosity were modeled, be boundary layer behavior would be much different, and there is no usefulness in modeling the suction without the viscosity. An 3D inviscid analysis could be done to validate the streamline locations by adding the displacement thicknesses computed by MISES to the blade surface.

The ultimate design validation would be to build the stage as described and test it. This could be done in a facility like the Blowdown Compressor at the MIT Gas Turbine Lab. Flowfield measurements that provided total temperature data would determine whether the fan really performs as designed. If the data showed that the flow was separated, that would imply that the suction was modeled incorrectly and it did not work as well as predicted in delaying separation of the boundary layer.

Appendix A

Source Code For Streamline Curvature Analysis

Main Program - main.f

```
      include 'vars.inc'

C sc.f begun 1/31/94, retyped 3/2/94
C By Larry Smilg
C This program calculates the axisymmetric throughflow though a fan.
C Entropy, Total Enthalpy, and geometric blockage
C must be defined in defs.f as functions of y and z

C Y is the radial direction and denoted by numr, and loop i
C Z is the axial direction and denoted by numm, and loop j

C The geometry is defined by the placement of the top and bottom
C streamlines.  They do not move.

C The velocity along a streamline is calculated by a streamwise
C equation of motion, and the radial (Y) position of the stream is
C computed by calculating conservation of mass between the streams.
C The Z position of each station does not move.

C Initialize variables and matrices
      call initial

      tol = 0.001

C When iterlim is set to zero, initial conditions can be examined
      if (iterlim .eq. 0) goto 20

 9    iter = 0
```

```
C Loop through the iterations
 10  continue
     iter = iter + 1

     errtot = 0.0

C Set up the old matrices for each position
     do j = 1,numm
       do i = 1,numr
         yold(i,j) = y(i,j)
         vmold(i,j) = vm(i,j)
       end do
     end do

C Compute inlet boundary

     call inlbc

     do j = 2,numm-1

C Compute state variables
       do i = 1,numr
         call comstate
       end do

C Find Vm across the passage with discretized eqn of motion
       call findvm

C Adjust computed velocities to conserve mass, then adjust positions
C This procedure loops if necessary

       call adj

     end do

C Update exit boundary (nonreflective)

     call exitbc

C update y positions by the relax factor
     do i = 1,numr
       do j = 1,numm
         errtot = errtot+(y(i,j)-yold(i,j))**2
         y(i,j) = yold(i,j) + relax*(y(i,j)-yold(i,j))
       end do
     end do

     rmserr(iter) = sqrt(errtot / (numr*numm))
     tol = rmserr(iter)
     write(6,931)'RMS error:',rmserr(iter),' massflow:   ',
    x    mdotin,' at iteration ',iter
     if (iter.lt.iterlim) goto 10

 931  format(a,g15.6,a,f8.3,a,i4)
```

```
 20   continue

C Use GRAFIC to look at the data

      call output

      if (iterlim.gt. 0) goto 10

      write(6,*) 'BYE BYE!'

      end
```

Varable declarations - vars.inc

```
      implicit none

      integer maxr,maxm
      parameter(maxr=33,maxm=256)

      real*8 y(maxr,maxm),z(maxr,maxm)
      integer numr,numm
      common /grid / y,z,numr,numm

      real ytip,ttf,rosta,roend,ststa,stend,psrat,omega
      common/ fan / ytip,ttf,rosta,roend,ststa,stend,psrat,omega

      real rlossfac,slossfac,rthick,sthick,rhubtc,rtiptc,shubtc,stiptc
      integer nrot,nstat
      common /perf / rlossfac,slossfac,rthick,sthick,nrot,nstat,
     &     rhubtc,rtiptc,shubtc,stiptc

      real*8 mto(maxr,maxm),mm(maxr,maxm),mth(maxr,maxm)
      real*8 vto(maxr,maxm),vm(maxr,maxm),vth(maxr,maxm)
      real*8 t(maxr,maxm),tt(maxr,maxm),pt(maxr,maxm),p(maxr,maxm)
      real*8 rho(maxr,maxm),beta(maxr,maxm),a(maxr,maxm)
      common/ stat / mto,mm,mth,vto,vm,vth,t,tt,p,pt,rho,beta,a

      real*8 vthrel(maxr,maxm),vtorel(maxr,maxm),mtorel(maxr,maxm)
      real*8 mthrel(maxr,maxm),aastar(maxr,maxm),aainl(maxr,maxm)
      real*8 betarel(maxr,maxm),ptrel(maxr,maxm),ttrel(maxr,maxm)
      real*8 anormrel(maxr,maxm),htrel(maxr,maxm),aastinl(maxr,maxm)
      real*8 anorm(maxr,maxm),ar(maxr,maxm)
      common /relst/ vthrel,vtorel,mtorel,mthrel,aastar,betarel,
     &     ptrel,ttrel,aainl,anormrel,htrel,aastinl,anorm,ar

      real*8 phi(maxr,maxm)
      real*8 rcinv,rc,drdz,drdz2,dr,dz
      common /geom / rcinv,rc,phi,drdz,drdz2,dr,dz

      integer iter
      real*8 errtot,rmserr(10000)
      common/ errs / errtot,rmserr,iter
```

```
      real*8 tol,visc
      common/ crap / visc,tol

      real*8 mdotcalc,mdotstr(maxr-1),mdotin,dmdwmo,wmo
      common/ mass / mdotcalc,dmdwmo,wmo,mdotstr,mdotin

      real cp,r,g,patm,tatm,tcru,pcru,mcru,minlet
      common/ cond / cp,r,g,patm,tatm,tcru,pcru,mcru,minlet

      integer iterlim,i,j,k
      real relax
      common/ run / iterlim,relax,i,j,k

      real*8 yold(maxr,maxm),vmold(maxr,maxm)
      common/ old / yold,vmold

      logical masschk
      common/ chk / masschk

      real*8 x1,x2,x3,y1,y2,y3,xr,yr,rds
      common/ circ / x1,x2,x3,y1,y2,y3,xr,yr,rds

      integer plottype,indgr,ncont
      real cont(200)
      common/ plot / plottype,indgr,ncont,cont
```

Function definitions - fns.inc

```
          real*8 s,ht,w,bl,masscomp
```

Flowfield initialization - init.f

```
C Initialize variables and read in grid

      subroutine initial

      include 'vars.inc'
      include 'fns.inc'

      character*4 id
      character*12 grname,dfname
      real yy(maxr,maxm),zz(maxr,maxm)

C define some constants
      cp = 1003.0
      r = 287.0
      g = 1.4

C define inlet conditions
```

```
C Initialize the velocity, the geometries and the streams

      write(6,*)'Enter the four character file ID: '
      read(5,1000) id
 1000 format(a4)

      grname(1:4) = id
      grname(5:12) = 'grid.dat'
      dfname(1:4) = id
      dfname(5:12) = 'data.dat'

C Initialize GRAFIC
      call grinit(5,6,'Streamline Curvature Calculation:  '//id)

      open(3,file=dfname,status='old')

      read(3,*) numr,numm
      read(3,*) rosta,roend
      read(3,*) ststa,stend
      read(3,*) ttf,minlet
      read(3,*) omega
      read(3,*) patm,tatm
      read(3,*) mcru
      read(3,*) rlossfac,slossfac
      read(3,*) rhubtc,rtiptc
      read(3,*) shubtc,stiptc
      read(3,*) nrot,nstat
      read(3,*) plottype
      close(3)

      open(2,file=grname,status='old')
      do i = 1,numr
        do j = 1,numm
          read(2,*) z(i,j),y(i,j)
        end do
      end do
      close(2)

C define cruise static pressure and temperature
      pcru = patm*((1+(g-1)/2*mcru**2))**(g/(g-1))
      tcru = tatm*((1+(g-1)/2*mcru**2))

      write(6,*)

 50   format(A,F8.4)
      dr = abs(y(1,1)-y(numr,1))
      write(6,50)'dr = ',dr
      dz = abs(z(1,numm)-z(1,1))/(numm-1)
      write(6,50)'dz = ',dz

      relax = 1.0/(1.0+.17*.36*dr**2/dz**2)
      write(6,50)'Optimum relax factor set at',relax
      write(6,*)
```

```
C Initialize state of matrix
      do j = 1,numm
        do i = 1,numr
          tt(i,j) = ht(yold(i,j),z(i,j))/cp
          pt(i,j) = pcru*(tt(i,j)/tcru)**(g/(g-1))*
     x        exp(-1.0*s(i,j)/r)
          if (j .le. rosta) then
C   flow before rotor - no swirl
            vth(i,j) = 0
          else if (j .le. roend) then
C   flow in rotor - Apply euler eqn
            vth(i,j) = cp*(tt(i,j)-tt(i,1))/(omega*y(i,j))
          else if (j.le.ststa) then
C   flow between rotor & stator - ang. mom. is same as rotor outlet
            vth(i,j) = vth(i,roend)*y(i,roend)/y(i,j)
          else if (j.le.stend) then
C   flow in stator - vth decreases linearly
            vth(i,j) = vth(i,roend)*(1.0-w(z(i,ststa),
     &           z(i,j),z(i,stend)))
          else
C   flow after stator - axial
            vth(i,j) = 0
          end if

          mth(i,j) = vth(i,j)/sqrt(g*r*275.0)

          mm(i,j) = minlet
          mto(i,j) = sqrt(mm(i,j)**2+mth(i,j)**2)

          t(i,j) = tt(i,j) / (1+(g-1)/2*mto(i,j)**2)
          p(i,j) = pt(i,j) / (1+(g-1)/2*mto(i,j)**2)**(g/(g-1))
          a(i,j) = sqrt(g*r*t(i,j))
          vto(i,j) = a(i,j)*mto(i,j)
          vm(i,j) = a(i,j)*mm(i,j)
          beta(i,j) = atan(vth(i,j)/vm(i,j))
          rho(i,j) = p(i,j)/(r*t(i,j))
          vthrel(i,j) = omega*y(i,j)-vth(i,j)
          vtorel(i,j) = sqrt(vm(i,j)**2+vthrel(i,j)**2)
          mtorel(i,j) = vtorel(i,j)/a(i,j)
          htrel(i,j) = ht(y(i,j),z(i,j))+vtorel(i,j)**2/2.0
          betarel(i,j) = atan(vthrel(i,j)/vm(i,j))
          beta(i,j) = atan(vth(i,j)/vm(i,j))
        end do
      end do

C Compute the initial mdot.  This is conserved down the stream
C This assumes straight flow at constant speed and density

      mdotin = 0.0
      j = 1
      do i = 2,numr
        mdotin = mdotin + masscomp(i-1,i)
      end do
```

```
      write(6,50)'Initial mdot in:   ',mdotin

      write(6,*)

      write(6,*)' Enter number of iterations:'
      read(5,*) iterlim

      return
      end
```

State computations - state.f

```
C Apply the streamline curvature equation to compute vm variation
C This is done in relative coordinates inside the rotor
C This works from the center streamline velocity outward to the hub
C and tip.  The center streamline velocity does not change.

      subroutine findvm

      include 'vars.inc'
      include 'fns.inc'

      real*8 dhtdr,tdsdr,acc,cent,swirl,change,dm,rotfram
      integer aa,bb,center

      center = (numr+1)/2

      do i = center-1,1,-1
        aa = i+1
        bb = i

        dm = sqrt((y(i,j)-y(i,j-1))**2+(z(i,j)-z(i,j-1))**2)
     x        +sqrt((y(i,j+1)-y(i,j))**2+(z(i,j+1)-z(i,j))**2)
        dr = y(aa,j)-y(bb,j)

        dhtdr = (ht(y(aa,j),z(aa,j))-ht(y(bb,j),z(bb,j)))/dr
        swirl = ((y(aa,j)*vth(aa,j))**2-(y(bb,j)*vth(bb,j))**2)
     x        / (2*y(i,j)**2*dr)
        tdsdr = t(i,j) / dr * (s(aa,j)-s(bb,j))
        acc = vm(i,j)*sin(phi(i,j))/dm*(vm(i,j+1)-vm(i,j-1))
        cent = (vm(i,j)**2)*rcinv*cos(phi(i,j))

        change = dhtdr-tdsdr+acc+cent-swirl

        vm(i,j) = sqrt(vm(i+1,j)**2-2*change*dr)
C         vm(i,j) = vmold(i,j) + 1.0*(vm(i,j)-vmold(i,j))
      end do

      do i = center+1,numr
        aa = i
        bb = i-1

        dm = sqrt((y(i,j)-y(i,j-1))**2+(z(i,j)-z(i,j-1))**2)
```

```
  x       +sqrt((y(i,j+1)-y(i,j))**2+(z(i,j+1)-z(i,j))**2)
         dr = y(aa,j)-y(bb,j)

         dhtdr = (ht(y(aa,j),z(aa,j))-ht(y(bb,j),z(bb,j)))/dr
         swirl = ((y(aa,j)*vth(aa,j))**2-(y(bb,j)*vth(bb,j))**2)
  x       / (2*y(i,j)**2*dr)
         tdsdr = t(i,j) / dr * (s(aa,j)-s(bb,j))
         acc = vm(i,j)*sin(phi(i,j))/dm*(vm(i,j+1)-vm(i,j-1))
         cent = (vm(i,j)**2)*rcinv*cos(phi(i,j))

         change = dhtdr-tdsdr+acc+cent-swirl

         vm(i,j) = sqrt(vm(i-1,j)**2+2*change*dr)
C          vm(i,j) = vmold(i,j) + 1.0*(vm(i,j)-vmold(i,j))
      end do

      end

C----

C Compute the new values of the thermodynamic state variables

      subroutine comstate

      include 'vars.inc'
      include 'fns.inc'

      real*8 mtp,ainl,height,htinl,astar,aast,astarinl
      real*8 mguess,aastmg,dadm,mng
      real*8 spacing,soffset,block

      tt(i,j) = ht(yold(i,j),z(i,j))/cp
 31   format(a,f9.3)
 32   format(a,f9.3,f9.3)

      if ((j.le.roend).and.(j.ge.rosta)) then

         ttrel(i,j) = tt(i,1)+omega**2*y(i,j)**2/(2.0*cp)

         vth(i,j) = cp*(tt(i,j)-tt(i,1))/(omega*y(i,j))
         vto(i,j) = sqrt(vth(i,j)**2+vm(i,j)**2)
         vthrel(i,j) = vth(i,j) - omega*y(i,j)
         vtorel(i,j) = sqrt(vthrel(i,j)**2+vm(i,j)**2)

         t(i,j) = ttrel(i,j)-vtorel(i,j)**2/(2.0*cp)
         a(i,j) = sqrt(g*r*t(i,j))

         mthrel(i,j) = vthrel(i,j)/a(i,j)
         mth(i,j) = vth(i,j)/a(i,j)
         mto(i,j) = vto(i,j)/a(i,j)
         mtorel(i,j) = vtorel(i,j)/a(i,j)
         mm(i,j) = vm(i,j)/a(i,j)

         ptrel(i,j) = pcru*(ttrel(i,j)/tcru)**(g/(g-1))*
```

```
     &        exp(-1.0*s(i,j)/r)
        pt(i,j) = pcru*(tt(i,j)/tcru)**(g/(g-1))*
     &        exp(-1.0*s(i,j)/r)
        p(i,j) = ptrel(i,j) / (1+(g-1)/2*mtorel(i,j)**2)**(g/(g-1))
      else
C Flow is not in the rotor

        if (j .le. rosta) then
C    flow before rotor - no swirl
          vth(i,j) = 0
C         else if (j .le. roend) then
C    flow in rotor - Apply euler eqn
C            vth(i,j) = cp*(tt(i,j)-tt(i,1))/(omega*y(i,j))
        else if (j.le.ststa) then
C    flow between rotor & stator - ang. mom. is same as rotor outlet
          vth(i,j) = vth(i,roend)*y(i,roend)/y(i,j)
        else if (j.le.stend) then
C    flow in stator - vth decreases linearly
          vth(i,j) = vth(i,roend)*(1.0-w(z(i,ststa),
     &           z(i,j),z(i,stend)))
        else
C    flow after stator - axial
          vth(i,j) = 0
        end if
        vto(i,j) = sqrt(vth(i,j)**2+vm(i,j)**2)

        pt(i,j) = pcru*(tt(i,j)/tcru)**(g/(g-1))*exp(-1.0*s(i,j)/r)
        t(i,j) = tt(i,j)-vto(i,j)**2/(2.0*cp)
        a(i,j) = sqrt(g*r*t(i,j))
        mto(i,j) = vto(i,j)/a(i,j)
        mth(i,j) = vth(i,j)/a(i,j)
        mm(i,j) = vm(i,j)/a(i,j)
        p(i,j) = pt(i,j) / (1+(g-1)/2*mto(i,j)**2)**(g/(g-1))
        mthrel(i,j) = mth(i,j) - omega*y(i,j)/a(i,j)
        mtorel(i,j) = sqrt(mthrel(i,j)**2+mm(i,j)**2)
      end if

      rho(i,j) = p(i,j)/(r*t(i,j))
      beta(i,j) = atan(vth(i,j)/vm(i,j))
      betarel(i,j) = atan(mthrel(i,j)/mm(i,j))

      drdz = 0.5*((yold(i,j+1)-yold(i,j))/(z(i,j+1)-z(i,j))+
     x     (yold(i,j)-yold(i,j-1))/(z(i,j)-z(i,j-1)))
      drdz2 = 2.0/(z(i,j+1)-z(i,j-1))*((yold(i,j+1)-yold(i,j)) /
     x     (z(i,j+1)-z(i,j)) - (yold(i,j)-yold(i,j-1)) /
     x     (z(i,j)-z(i,j-1)))

      phi(i,j) = atan(drdz)

      if (i.eq.1) then
        ar(i,j) = 3.14159*(y(2,j)**2-y(1,j)**2)*bl(y(i,j),z(i,j))
      else if (i.eq.numr) then
        ar(i,j) = 3.14159*(y(i,j)**2-y(i-1,j)**2)*bl(y(i,j),z(i,j))
      else
```

```
         ar(i,j) = 3.14159*(y(i+1,j)**2-y(i-1,j)**2)*bl(y(i,j),z(i,j))
      end if

      anormrel(i,j) = ar(i,j)*cos(betarel(i,j))*cos(phi(i,j))
      anorm(i,j) = ar(i,j)*cos(phi(i,j))

      rcinv = drdz2 / (1.0 + drdz**2.0)**1.5

      end
C-----
      real*8 function masscomp(i1,i2)

      include 'vars.inc'

      integer i1,i2
      real*8 bl
      real*8 rhoav,phiav,vmav,yav,ds,ma

      rhoav = (rho(i1,j)+rho(i2,j))/2.0
      phiav = (phi(i1,j)+phi(i2,j))/2.0
      vmav = (vm(i1,j)+vm(i2,j))/2.0
      yav = (y(i1,j)+y(i2,j))/2.0

      ds = bl(yav,z(i1,j))*3.14159*cos(phiav)*(y(i2,j)**2-y(i1,j)**2)

      ma = rhoav*vmav*ds
      masscomp = ma

      return
      end
```

Streamline position and velocity adjustment - adj.f

```
      subroutine adj

C adjust all streamline velcities to conserve overall mass,
C then adjust streamline positions to conserve streamtube mass

      real*8 rhoav,phiav,vmav,mmav,yav
      real*8 scale,ds,oldscale,screl

      include 'vars.inc'
      include 'fns.inc'

      oldscale = 5.0
      screl = 1.0

 30   continue

      wmo = vm((numr+1)/2,j)
      mdotcalc = 0.0
      dmdwmo = 0.0
```

```
      if(y(numr-1,j).gt.y(numr,j)) then
C scale velocities to put y values back into duct
        scale = 0.1
      else
C Sum mdot and d(mdot)/d(wmo)
        do i = 2,numr
          rhoav = (rho(i,j)+rho(i-1,j))/2.0
          vmav = (vm(i,j)+vm(i-1,j))/2.0
          mmav = (mm(i,j)+mm(i-1,j))/2.0
          yav = (y(i,j)+y(i-1,j))/2.0
          ds = bl(yav,z(i,j))*3.14159*cos(phiav)*
     &        (y(i,j)**2-y(i-1,j)**2)
          mdotcalc = mdotcalc+masscomp(i-1,i)
          dmdwmo = dmdwmo + rhoav*(1.0-mmav**2)*vmav/wmo*ds
        end do
        scale = (mdotin-mdotcalc)/(wmo*dmdwmo)

        if (scale.lt.-1.0) scale = -0.5
        if(abs(scale).gt.abs(oldscale)) screl = 0.5*screl

      end if

      do i = 1,numr
        vm(i,j) = vm(i,j)*(1.0+scale*screl)

C Find new state variables with change in vm
        vto(i,j) = sqrt(vm(i,j)**2+vth(i,j)**2)

        if ((j.le.roend).and.(j.ge.rosta)) then
          vtorel(i,j) = sqrt(vm(i,j)**2+vthrel(i,j)**2)
          t(i,j) = ttrel(i,j)-vtorel(i,j)**2/(2.0*cp)
          a(i,j) = sqrt(g*r*t(i,j))
          mtorel(i,j) = vtorel(i,j)/a(i,j)
          mto(i,j) = vto(i,j)/a(i,j)
          p(i,j) = ptrel(i,j)/(1.0+(g-1.0)/2.0*
     &        mtorel(i,j)**2)**(g/(g-1.0))
        else
          t(i,j) = tt(i,j)-vto(i,j)**2/(2.0*cp)
          a(i,j) = sqrt(g*r*t(i,j))
          vtorel(i,j) = sqrt(vm(i,j)**2+vthrel(i,j)**2)
          mto(i,j) = vto(i,j)/a(i,j)
          mtorel(i,j) = vtorel(i,j)/a(i,j)
          p(i,j) = pt(i,j)/(1.0+(g-1.0)/2.0*
     &        mto(i,j)**2)**(g/(g-1.0))
        end if

        mth(i,j) = vth(i,j)/a(i,j)
        mthrel(i,j) = vthrel(i,j)/a(i,j)
        mm(i,j) = vm(i,j)/a(i,j)
        rho(i,j) = p(i,j)/(r*t(i,j))
        beta(i,j) = atan(vth(i,j)/vm(i,j))
        betarel(i,j) = atan(vthrel(i,j)/vm(i,j))
      end do
```

```
      if((abs(scale).gt. tol).and.(y(numr,j).gt.y(numr-1,j))) goto 30

C Adjust the streamline positions by computing the mass
C flowing between them

      do i = 2,numr-1
        rhoav = (rho(i,j)+rho(i-1,j))/2.0
        vmav = (vm(i,j)+vm(i-1,j))/2.0
        phiav = (phi(i,j)+phi(i-1,j))/2.0
        yav = (y(i,j)+y(i-1,j))/2.0

        y(i,j) = sqrt(y(i-1,j)**2+mdotstr(i-1)/(3.14159*rhoav
     &        *cos(phiav)*vmav*bl(yav,z(i,j))))
      end do

C Check mass across the modified streamtubes

      mdotcalc = 0.0
      do i = 2,numr
        mdotcalc = mdotcalc+masscomp(i-1,i)
      end do

      if(abs((mdotcalc-mdotin)/mdotin).gt.tol)
     &    goto 30

      end
```

Boundary computations - bcs.f

```
C This routine applies nonreflective BCs at the inlet instead of
C enforcing uniform values

      subroutine inlbc

      include 'vars.inc'
      include 'fns.inc'

      real*8 rhoav,phiav,vmav,mmav,yav,dm
      real*8 scale,ds,desmdot

      do i = 1,numr

C The inlet mach number is given, and there is no pre-swirl
        mth(i,1) = 0
        mto(i,1) = minlet
        p(i,1) = pt(i,1) / (1+(g-1)/2*mto(i,1)**2)**(g/(g-1))

        tt(i,1) = ht(yold(i,1),z(i,1))/cp
        pt(i,1) = pcru*(tt(i,1)/tcru)**(g/(g-1))*
     x      exp(-1.0*s(i,1)/r)
```

```
        mm(i,1) = sqrt(mto(i,1)**2-mth(i,1)**2)
        t(i,1) = tt(i,1) / (1+(g-1)/2*mto(i,1)**2)
        a(i,1) = sqrt(g*r*t(i,1))
        vto(i,1) = a(i,1)*mto(i,1)
        vth(i,1) = a(i,1)*mth(i,1)
        vm(i,1) = a(i,1)*mm(i,1)
        beta(i,1) = atan(vth(i,1)/vm(i,1))
        rho(i,1) = p(i,1)/(r*t(i,1))

        drdz = (yold(i,2)-yold(i,1))/(z(i,2)-z(i,1))
        phi(i,1) = atan(drdz)

      end do

C Compute the mass flowing into the duct

      mdotin = 0.0
      j = 1

      do k = 1,numr-1
        mdotstr(k) = masscomp(k,k+1)
        mdotin = mdotin+mdotstr(k)
      end do

      desmdot = mdotin/(numr-1.0)

      do i = 2,numr-1
        rhoav = (rho(i,1)+rho(i-1,1))/2.0
        phiav = (phi(i,1)+phi(i-1,1))/2.0
        vmav = (vm(i,1)+vm(i-1,1))/2.0
        yav = (y(i,1)+y(i-1,1))/2.0

        y(i,1) = sqrt(y(i-1,1)**2+desmdot / (3.14159*rhoav
     x       *cos(phiav)*vmav*bl(yav,z(i,1))))

      end do

C Recheck mass

      mdotin = 0.0

      do k = 1,numr-1
        mdotstr(k) = masscomp(k,k+1)
        mdotin = mdotin+mdotstr(k)
      end do

      end
C-----

C Compute the exit conditions

      subroutine exitbc

      include 'vars.inc'
```

```
      include 'fns.inc'

      real*8 rhoav,phiav,vmav,mmav,yav
      real*8 scale,ds,oldscale,screl

      do i = 1,numr

C Assume unchanging static pressure and no exit swirl
C then compute state

         mth(i,numm) = 0
         p(i,numm) = p(i,numm-1)
         tt(i,numm) = ht(yold(i,numm),z(i,numm))/cp
         pt(i,numm) = pcru*(tt(i,numm)/tcru)**(g/(g-1))*
     x        exp(-1.0*s(i,numm)/r)

         mto(i,numm) = (2.0/(g-1.0)*((pt(i,numm)/p(i,numm))
     x        **((g-1)/g)-1.0))**0.5
         mm(i,numm) = sqrt(mto(i,numm)**2-mth(i,numm)**2)
         t(i,numm) = tt(i,numm)/(1+(g-1)/2*mto(i,numm)**2)

         a(i,numm) = sqrt(g*r*t(i,numm))
         vto(i,numm) = a(i,numm)*mto(i,numm)
         vth(i,numm) = a(i,numm)*mth(i,numm)
         vm(i,numm) = a(i,numm)*mm(i,numm)
         beta(i,numm) = atan(vth(i,numm)/vm(i,numm))
         rho(i,numm) = p(i,numm)/(r*t(i,numm))

         drdz = (yold(i,numm)-yold(i,numm-1))/(z(i,numm)-z(i,numm-1))

         phi(i,numm) = atan(drdz)

       end do

C Adjust streamline velocities to conserve overall mass,
C then adjust stream positions to conserve streamtube mass.


       oldscale = 5.0
       screl = 1.0

       j = numm

 30    continue

       wmo = vm((numr+1)/2,numm)
       mdotcalc = 0.0
       dmdwmo = 0.0

C      Integrate to find mdotcalc and dmdwmo

       j = numm

       if (y(numr-1,j).gt.y(numr,j)) then
```

```
C scale velocities to put y values back inside the duct
         scale = 0.01
      else
 C    Integrate to find mdotcalc and dmdwmo

 C Sum mdot and d(mdot)/d(wmo)
         do i = 2,numr
            rhoav = (rho(i-1,j)+rho(i,j))/2.0
            mmav = (mm(i-1,j)+mm(i,j))/2.0
            vmav = (vm(i-1,j)+vm(i,j))/2.0
            phiav = (phi(i-1,j)+phi(i,j))/2.0
            yav = (y(i,j)+y(i-1,j))/2.0

            ds = bl(yav,z(i-1,j))*3.14159*cos(phiav)*
     &           (y(i,j)**2-y(i-1,j)**2)

            mdotcalc = mdotcalc+masscomp(i-1,i)
 C    write(6,*) mdotcalc

            dmdwmo = dmdwmo + rhoav*(1.0-mmav**2)*vmav/wmo*ds

         end do

         scale = (mdotin-mdotcalc)/(wmo*dmdwmo)
      end if

      if (abs(scale).gt.abs(oldscale)) then
         screl = 0.5*screl
         if (screl.lt. 0.0001) screl = 0.5
         write(6,*)'Relaxing scale factor in exit.'
         write(6,*)'sc rel',scale,screl
         write(6,*)'mdi mdc',mdotin,mdotcalc
      endif

      if (mdotcalc.lt.0) scale = -1.0*scale
      if (scale.le.-1.0) scale = -0.5

      do i = 1,numr
         vm(i,numm) = vm(i,numm)*(1.0+scale*screl)
      end do

      oldscale = scale

      if ((abs(scale) .gt. tol).and.(y(numr,j).gt.y(numr-1,j))) goto 30

C Adjust the streamline positions by computing the mass
C flowing between them.

      do i = 2,numr-1
         rhoav = (rho(i,j)*y(i,j)+rho(i-1,j)*y(i-1,j))/
     x        (y(i,j)+y(i-1,j))
         phiav = (phi(i,j)*y(i,j)+phi(i-1,j)*y(i-1,j))/
     x        (y(i,j)+y(i-1,j))
         vmav = (vm(i,j)*y(i,j)+vm(i-1,j)*y(i-1,j))/
```

```
     x        (y(i,j)+y(i-1,j))
        yav = (y(i-1,j)**2+y(i,j)**2)/(y(i-1,j)+y(i,j))

        y(i,j) = sqrt(y(i-1,j)**2+mdotstr(i-1) / (3.14159*rhoav
     x         *cos(phiav)*vmav*bl(yav,z(i,j))))

      end do

C Check mass in the modified streamtubes

      mdotcalc=0.0

      do i = 2,numr
        mdotcalc = mdotcalc+masscomp(i-1,i)
      end do

      if (abs((mdotcalc-mdotin)/mdotin) .gt. tol) goto 30

      end

C-----
```

State input definitions - defs.f

```
C Define these functions to give the geometry of the blade

C--------------------

C Entropy function
C entropy must be defined as zero at the beginning of each streamline

      real*8 function s(ii,jj)

      include 'vars.inc'

      integer ii,jj
      real*8 mtip,mtang,mbp2,mc2
      real*8 rloss,sloss,w

      mtang = omega*y(ii,jj)/a(ii,jj)
      mbp2 = mm(ii,rosta)**2 + mtang**2
      mc2 = mto(ii,ststa)**2

      rloss = -1.0*r*log(1-rlossfac*(1-(1+(g-1)/2*mbp2)**(g/(1-g))))
      sloss = -1.0*r*log(1-slossfac*(1-(1+(g-1)/2*mbp2)**(g/(1-g))))

      if (jj .le. rosta) then
C flow before rotor
        s = 0
      else if (jj .le. roend) then
C flow in rotor - Apply euler eqn
        s = rloss*w(z(ii,rosta),z(ii,jj),z(ii,roend))
      else if (jj .le. ststa) then
```

```
C flow between rotor & stator
        s = rloss
      else
C flow in stator (or after)
        s = rloss+sloss*w(z(ii,ststa),z(ii,jj),z(ii,stend))
      end if

      return
      end

C----------------------------
C total enthalpy function

      real*8 function ht(yy,zz)

      include 'vars.inc'

      real*8 yy,zz,htb,w

      htb = cp*tcru*(1+(g-1)/2*mcru**2)
      ht = htb*(1+w(z(i,rosta),zz,z(i,roend))*(ttf-1))

      return
      end
C------------------------------
C "Work" function called by other functions
C gives fraction of distance (0-1) between two points

      real*8 function w(p1,p2,p3)

      include 'vars.inc'

      real*8 p1,p2,p3

      if (p2.lt.p1) then
        w = 0.0
      else if (p2.gt.p3) then
        w = 1.0
      else
        w = (p2-p1)/(p3-p1)
      end if

      return
      end

C------------------------------
C Blockage function (0-1 where 1 is completely open)
      real*8 function bl(yy,zz)

      include 'vars.inc'

      real*8 yy,zz,thick,circum,block,w
      real*8 rch,sch,b1,b2,dx,x0,y0,xi,yi,sta
```

```
C Blockage from wakes assumed to be .05 of max blade thickness.
C Loss model should account for the mixed out wakes
      b1 = --1.0*betarel(i,rosta)
      b2 = -1.0*betarel(i,roend)
      dx = sqrt((z(i,roend)-z(i,rosta))**2+
     &     (y(i,roend)-y(i,rosta))**2)
      x0 = -1.0*dx*sin(b1)/(sin(b1)-sin(b2))
      y0 = -1.0/tan(b1)*x0
      xi = x0+dx
      yi = -1.0/tan(b2)*xi
      sta = atan((yi-y0)/(xi-x0))
      rch = dx/cos(sta)
      rthick = rch*(rhubtc+(rtiptc-rhubtc)*
     &     w(y(1,rosta),y(i,rosta),y(numr,rosta)))

      b1 = beta(i,ststa)
      b2 = beta(i,stend)
      dx = sqrt((z(i,stend)-z(i,ststa))**2+
     &     (y(i,stend)-y(i,ststa))**2)
      x0 = -1.0*dx*sin(b1)/(sin(b1)-sin(b2))
      y0 = -1.0/tan(b1)*x0
      xi = x0+dx
      yi = sqrt(x0**2+y0**2)

      sta = atan((yi-y0)/(xi-x0))
      sch = dx/cos(sta)
      sthick = sch*(shubtc+(stiptc-shubtc)*
     &     w(y(1,ststa),y(i,ststa),y(numr,ststa)))

      if (j .le. rosta) then
C flow before rotor
         thick = 0
      else if (j .le. roend) then
C flow in rotor
         thick = rthick*nrot*(1.0-2.0*abs(0.5-
     &        w(z(i,rosta),zz,z(i,roend)))+.05*w(
     &        z(i,rosta),zz,z(i,roend)))
      else if (j.le.ststa) then
C flow between rotor & stator
         thick = 0.05*rthick*nrot
      else if (j.le.stend) then
C flow in stator
         thick = 0.05*rthick*nrot+sthick*nstat*
     &        (1.0-2.0*abs(0.5-w(z(i,ststa),zz,z(i,stend)))+.05*
     &        w(z(i,ststa),zz,z(i,stend)))
      else
C flow after stator
         thick = 0.05*(rthick*nrot+sthick*nstat)
      end if

      circum = 2*3.14159*yy

      if (circum.eq.0) circum = 1000000.0
      block = 1.0 - thick/circum
```

```
      if (block.lt.0.0) then
        write(5,*)'blockage error!  bl was ',block
        write(5,*)'bl set to 0.1 at i j',i,j
        block = 0.1
      end if

      bl = block

      return
      end

C----
```

Plotting routine - output.f

```
C This calls the GRAFIC routine for output

      subroutine output
      include 'vars.inc'
      include 'fns.inc'

      integer ans,key,gd,lins,aa,bb
      real rms(10000),ints(10000),var(maxr,maxm)
      real zz(maxr,maxm),yy(maxr,maxm),st(maxr,maxm)
      real mstr(maxr-1),div
      character*50 title

C Convert grid doubles to reals for GRAFIC.  This is also done to the
C state variables in coplot

      do i = 1,numr
        do j = 1,numm
          zz(i,j) = z(i,j)
          yy(i,j) = y(i,j)
        end do
      end do

      ncont = 25
      indgr = 23

 10   write(6,*)
      write(6,*)' Choose number of choice:'
      write(6,*)'0.  Exit Program'
      write(6,*)'1.  Run more iterations'
      write(6,*)'2.  Change plot type'
      write(6,*)'3.  Change relaxation factor'
      write(6,*)'4.  Save camber lines'
      write(6,*)' Look at data for:'
      write(6,*)'5.  RMS error (convergence history)'
      write(6,*)'6.  Velocity triangles and flow path'
      write(6,*)'7.  Final Grid'
```

```
      write(6,*)'8.  Streamwise Velocity'
      write(6,*)'9.  Streamwise Mach Number'
      write(6,*)'10.  Swirl Velocity'
      write(6,*)'11.  Total Velocity'
      write(6,*)'12.  Total Mach number'
      write(6,*)'13.  Axial Flow Angle (phi)'
      write(6,*)'14.  Swirl Flow Angle (beta)'
      write(6,*)'15.  Density'
      write(6,*)'16.  Total Pressure'
      write(6,*)'17.  Static Pressure'
      write(6,*)'18.  Total Enthalpy'
      write(6,*)'19.  Static Temperature'
      write(6,*)'20.  Entropy'
      write(6,*)'21.  Blockage Factor'
      write(6,*)'22.  Relative Total Enthalpy'
      write(6,*)'23.  Relative Total Mach number'
      write(6,*)'24.  Relative Swirl Mach number '
      write(6,*)'25.  Relative Total Velocity'
      write(6,*)'26.  Relative Swirl Velocity'
      write(6,*)'27.  Relative Total Temperature'
      write(6,*)'28.  Relative Total Pressure'
      write(6,*)'29.  Relative Swirl Angle'
      write(6,*)'30.  Streamwise Flow Area (relative flow)'
      read(5,*) ans

      if (ans.eq.0) goto 100

      goto (122,121,120,118,119,117,101,102,103,104,105,106,107,108,
     x     109,110,111,112,113,114,116,123,124,125,126,127,
     x     128,129,130,131) (ans)
      write(6,*)'Invalid Choice.  Choose 0-30 only.'
      goto 10

100   iterlim = 0
      goto 500

101   call grgrid(zz,yy,numr,numm,maxr,maxm,'~z~r~Grid',indgr)
      goto 10

102   title = '~z~r~Velocity contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = vm(i,j)
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

103   title = '~z~r~Streamwise Mach Number contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = mm(i,j)
        end do
      end do
```

```
      call coplot(st,title,zz,yy)
      goto 10

104   title = '~z~r~Swirl Velocity contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = vth(i,j)
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

105   title = '~z~r~Total Velocity contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = vto(i,j)
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

106   title = '~z~r~Total Mach Number contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = mto(i,j)
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

107   title = '~z~r~Flow Angle (phi) contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = phi(i,j)*180.0/3.14159
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

108   title = '~z~r~Swirl Flow Angle (beta) contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = 180.0/3.14159*beta(i,j)
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

109   title = '~z~r~Density  contours'
      do i= 1,numr
        do j = 1,numm
          st(i,j) = rho(i,j)
        end do
      end do
```

```
    call coplot(st,title,zz,yy)
    goto 10

110 title = '~z~r~Total Pressure contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = pt(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

111 title = '~z~r~Static Pressure contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = p(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

112 title = '~z~r~Total Enthalpy contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = ht(y(i,j),z(i,j))
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

113 title = '~z~r~Static Temperature contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = t(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

114 title = '~z~r~Entropy contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = s(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

116 title = '~z~r~Blockage contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = bl(y(i,j),z(i,j))
      end do
    end do
```

```
      call coplot(st,title,zz,yy)
      goto 10

117   call vtri
      goto 10

118   call savecam
      goto 10

119   write(6,*)'Convergence History'
      do i = 1,iter
         ints(i) = i*1.0
         rms(i) = log10(rmserr(i))
      end do

      call grline(1,0,1,'~Iteration~log RMS error~Convergence History',
     x     21,ints,rms,iter)
      goto 10

120   write(6,*)'Old relaxation factor was:  ',relax
      write(6,*)'Enter new relaxation factor:'
      read(5,*) relax
      goto 10

121   write(6,*)' '
      write(6,*)'Change plot parameters'
      write(6,*)'1.  Contour plot style (color/grey/line):  PLOTTYPE= '
     x   , plottype
      write(6,*)'2.  Number of contours:  NCONT = ',ncont
      write(6,*)'3.  Change INDGR: INDGR = ',indgr
      write(6,*)'4.  return to plot menu'
      read(5,*) ans

      goto (201,202,203,200) (ans)
      write(6,*) 'Invalid Choice.    Choose 1-4 only'
      goto 121

200   goto 10

201   write(6,*)' '
      write(6,*)'Enter plot type'
      write(6,*)'1.  Color'
      write(6,*)'2.  Greyscale'
      write(6,*)'3.  B/W lines'
      write(6,*)'4.  Return to parameter menu'
      read(5,*) ans

      if (ans .eq. 4) goto 121

      goto (210,210,210) (ans)

      write(6,*)'Invalid Choice.  choose 1-4 only'
      goto 201
```

```
210 write(6,*)' '
    write(6,*)'0.  No key '
    write(6,*)'1.  Key '
    read(5,*) key

    write(6,*)' '
    write(6,*)'0.  No grid '
    write(6,*)'1.  Superimpose grid '
    read(5,*) gd

    if (ans .eq. 3) goto 303

    write(6,*)' '
    write(6,*)'0.  No lines'
    write(6,*)'1.  Superimpose contour lines '
    read(5,*) lins

    if (ans .eq. 2) goto 302

301 call grinit(5,6,'Streamline Curvature calculation')
    plottype = 2+gd*4+key*8+lins
    goto 121

302 call grgrey
    plottype = 2+gd*4+key*8+lins
    goto 121

303 plottype = 1+gd*4+key*8
    goto 121

202 write(6,*)'Enter new number of contours:'
    read(5,*) ncont
    goto 121

203 write(6,*)'enter new value of INDGR'
    read(5,*) indgr
    goto 121

122 write(6,*)'Enter number of additional iterations:'
    read(5,*) ans
    iterlim = iterlim + ans
    goto 500

123 title = '~z~r~Total Relative Enthalpy contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = htrel(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

124 title = '~z~r~Total Relative Mach contours'
    do i= 1,numr
```

```
      do j = 1,numm
        st(i,j) = mtorel(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

125 title = '~z~r~Swirl Relative Mach contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = mthrel(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

126 title = '~z~r~Total Relative velocity contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = vtorel(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

127 title = '~z~r~Swirl Relative velocity contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = vthrel(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

128 title = '~z~r~Relative Total Temperature contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = ttrel(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

129 title = '~z~r~Relative Total Pressure contours'
    do i= 1,numr
      do j = 1,numm
        st(i,j) = ptrel(i,j)
      end do
    end do
    call coplot(st,title,zz,yy)
    goto 10

130 title = '~z~r~Relative Swirl Flow Angle (beta) contours'
    do i= 1,numr
```

```
        do j = 1,numm
          st(i,j) = 180.0/3.14159*betarel(i,j)
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

 131  title = '~z~r~Streamwise Area contours'
      do i= 1,numr
        do j = 1,numm
          if ((j.le.roend) .and. (j.ge.rosta)) then
            st(i,j) = anormrel(i,j)/anormrel(i,rosta)
          else
            st(i,j) = anorm(i,j)/anormrel(i,rosta)
          end if
        end do
      end do
      call coplot(st,title,zz,yy)
      goto 10

 134  call ochoke
      goto 10

 500  continue
      end

C----------------------------------

      subroutine coplot(state,title,zz,yy)
      include 'vars.inc'

C Make a contour plot using the state variable given

      real state(maxr,maxm),zz(maxr,maxm),yy(maxr,maxm)
      character*40 title

      write(6,*) title(6:30)

      call grcfil(state,numr,numm,maxr,maxm,ncont,cont)
      call grcont(zz,yy,state,numr,numm,maxr,maxm,title,
     x     indgr,cont,ncont,plottype)

      end
C------

      subroutine ochoke
      include 'vars.inc'
      include 'fns.inc'

      real*8 amin,ainl,aex,at
      real*8 b1,b2,h1,h2
      real*8 dinl,dt,dex
      real*8 minl,xt,aast
      real*8 rle,ch,dx,spc,thick
```

```
      real*8 rd,bmt

      write(6,*) 'Stream   di     dt     do     ai     at     ao     amin
     &  b1     b2     bmt'

      rd = 180.0/3.141593

      do i = 1,numr

        b1 = -1.0*betarel(i,rosta)
        b2 = -1.0*betarel(i,roend)

        if (i.eq.1) then
          h1 = y(2,rosta)-y(1,rosta)
          h2 = y(2,roend)-y(1,roend)
        else if (i.eq.numr) then
          h1 = y(numr,rosta)-y(numr-1,rosta)
          h2 = y(numr,roend)-y(numr-1,roend)
        else
          h1 = y(i+1,rosta)-y(i-1,rosta)
          h2 = y(i+1,roend)-y(i-1,roend)
        end if

        dx = sqrt((z(i,roend)-z(i,rosta))**2+
     &        (y(i,roend)-y(i,rosta))**2)
        spc = 2*3.14159*y(i,rosta)/nrot

        thick = (rhubtc+(rtiptc-rhubtc)*
     &    w(y(1,rosta),y(i,rosta),y(numr,rosta)))
        ch = dx/cos((b1+b2)/2.0)
        rle = thick*ch*.005

        xt = spc/2.0*sin(2.0*b1)
        minl = mtorel(i,rosta)
        aast = 1.0/minl*((1.0+0.2*minl**2.0)/1.2)**3.0
        ht = h1+(h2-h1)*w(0,xt,dx)

        dinl = spc*cos(b1)
        dt = dinl - rle
        dex = spc*cos(b2)

        ainl = dinl*h1
        at = dt*ht
        aex = dex*h2

        amin = ainl/aast

        bmt = acos(amin/(ht*spc))

13      format(i6,f7.4,f7.4,f7.4,f7.4,f7.4,f7.4,f7.4,f7.3,f7.3,f7.3)
        write(6,13) i,dinl,dt,dex,ainl,at,aex,amin,b1*rd,b2*rd,bmt*rd
      end do

      end
```

Velocity triangle generator - veltri.f

```
C Make velocity triangles and blade sketches for rotor and stator

      subroutine vtri
      include 'vars.inc'

      integer str,b,c,c2,d
      character*10 titl(6)
      integer ilin(6),isym(6),nper(6)
      real th(12),me(12)
      real drot,dstat,solrot,solstat,rch,sch
      real wb,wc,vb,vc,vbp,vcp,vd
      integer lpt,npts,rp
      real z1,z3,bp,bpo
      real zp(600),tp(600),sp(600)

 500  format(a,f7.3)

      b = rosta
      c = roend
      c2 = ststa
      d = stend

      write(6,*)'Enter the stream number for the triangles'
      read(5,*) str

C Draw rotor

      bpo = atan((vth(str,b)-omega*y(str,b))/vm(str,b))
      bp = atan((vth(str,b+1)-omega*y(str,b+1))/vm(str,b+1))

      zp(1) = z(str,b)
      tp(1) = 0
      sp(1) = 2*3.14159*y(str,b)/nrot

      lpt = 2
      do i = b,c-1
        z1 = z(str,i)
        z3 = z(str,i+1)
        do j = 1,9
          zp(lpt) = z(str,i)+j/10.0*(z(str,i+1)-z(str,i))
          tp(lpt) = tp(lpt-1)+tan(bpo+(zp(lpt)-z1)/(z3-z1)
     &           *(bp-bpo))*(zp(lpt)-zp(lpt-1))
          sp(lpt) = 2*3.14159*(y(str,i)+(zp(lpt)-z1)/(z3-z1)*
     &           (y(str,i+1)-y(str,i)))/nrot
          lpt = lpt + 1
        end do

        bpo = bp
        bp = atan((vth(str,i+2)-omega*y(str,i+2))/vm(str,i+2))
```

```
        zp(lpt) = z(str,i+1)
        tp(lpt) = tp(lpt-1)+tan(bpo)*(zp(lpt)-zp(lpt-1))
        sp(lpt) = 2*3.14159*y(str,i+1)/nrot
        lpt = lpt + 1
      end do

      npts = lpt-1

      do i = npts+1,2*npts
        zp(i) = zp(i-npts)
        tp(i) = tp(i-npts)+sp(i-npts)
        zp(i+npts) = zp(i)
        tp(i+npts) = tp(i-npts)-sp(i-npts)
      end do

      rch = sqrt((zp(npts)-zp(1))**2+(tp(npts)-tp(1))**2)

      ilin(1) = 1
      ilin(2) = 3
      ilin(3) = 3
      isym(1) = 0
      isym(2) = 0
      isym(3) = 0
      nper(1) = npts
      nper(2) = npts
      nper(3) = npts

      titl(1) = 'Rotor     '
      titl(2) = 'distance  '
      titl(3) = 'distance  '

      rp = 3*npts

C Now do Stator

      zp(rp+1) = z(str,c2)
      tp(rp+1) = tp(npts)
      sp(rp+1) = 2*3.14159*y(str,c2)/nstat

      lpt = 2
      do i = c2,d-1
        z1 = z(str,i)
        z3 = z(str,i+1)
        do j = 1,9
          zp(rp+lpt) = z(str,i)+j/10.0*(z(str,i+1)-z(str,i))
          tp(rp+lpt) = tp(rp+lpt-1)+tan(beta(str,i)+(zp(rp+lpt)-z1)
     &         /(z3-z1)*(beta(str,i+1)-beta(str,i)))*
     &         (zp(rp+lpt)-zp(rp+lpt-1))
          sp(rp+lpt) = 2*3.14159*(y(str,i)+(zp(rp+lpt)-z1)/(z3-z1)*
     &         (y(str,i+1)-y(str,i)))/nstat
          lpt = lpt + 1
        end do

        zp(rp+lpt) = z(str,i+1)
```

```
        tp(rp+lpt) = tp(rp+lpt-1)+tan(beta(str,i+1))*
     &       (zp(rp+lpt)-zp(rp+lpt-1))
        sp(rp+lpt) = 2*3.14159*y(str,i+1)/nstat
        lpt = lpt + 1
      end do

      npts = lpt-1

      sch = sqrt((zp(rp+npts)-zp(rp+1))**2+(tp(rp+npts)-tp(rp+1))**2)

      do i = npts+1,2*npts
        zp(rp+i) = zp(rp+i-npts)
        tp(rp+i) = tp(rp+i-npts)+sp(rp+i-npts)
        zp(rp+i+npts) = zp(rp+i)
        tp(rp+i+npts) = tp(rp+i-npts)-sp(rp+i-npts)
      end do

      ilin(4) = 2
      ilin(5) = 3
      ilin(6) = 3
      isym(4) = 0
      isym(5) = 0
      isym(6) = 0
      nper(4) = npts
      nper(5) = npts
      nper(6) = npts

      titl(4) = 'Stator    '
      titl(5) = 'distance  '
      titl(6) = 'distance  '

C Draw the blades
      call grklin(ilin,isym,nper,titl,6,zp,tp,'~m~m~Blade paths',23)

C Do the rotor triangle

C Vb
      nper(1) = 2
      ilin(1) = 2
      isym(1) = 0
      titl(1) = 'Vb        '
      th(1) = 0.0
      me(1) = 0.0
      th(2) = 0.0
      me(2) = vm(str,b)
      vb = sqrt(th(2)**2+me(2)**2)

C omega rb
      nper(2) = 2
      ilin(2) = 1
      isym(2) = 0
      titl(2) = 'wrb       '
      th(3) = 0.0
      me(3) = vm(str,b)
```

```
      th(4) = omega*y(str,b)
      me(4) = vm(str,b)
      wb = omega*y(str,b)

C Vb prime
      nper(3) = 2
      ilin(3) = 3
      isym(3) = 0
      titl(3) = 'Vb'          '
      th(5) = 0.0
      me(5) = 0.0
      th(6) = omega*y(str,b)
      me(6) = vm(str,b)
      vbp = sqrt(th(6)**2+me(6)**2)

C Vc
      nper(4) = 2
      ilin(4) = 4
      isym(4) = 0
      titl(4) = 'Vc           '
      th(7) = 0.0
      me(7) = 0.0
      th(8) = -1.0*vth(str,c)
      me(8) = vm(str,c)
      vc = sqrt(th(8)**2+me(8)**2)

C omega rc
      nper(5) = 2
      ilin(5) = 1
      isym(5) = 0
      titl(5) = 'wrc          '
      th(9) = -1.0*vth(str,c)+omega*y(str,c)
      me(9) = vm(str,c)
      th(10) = -1.0*vth(str,c)
      me(10) = vm(str,c)
      wc = omega*y(str,c)

C Vc prime
      nper(6) = 2
      ilin(6) = 5
      isym(6) = 0
      titl(6) = 'Vc'          '
      th(11) = 0.0
      me(11) = 0.0
      th(12) = omega*y(str,c)-vth(str,c)
      me(12) = vm(str,c)
      vcp = sqrt(th(12)**2+me(12)**2)

      solrot = rch / (2*3.14159*y(str,c)/nrot)
      drot = 1 - vcp/vbp + abs(vth(str,c)*y(str,c)-
     &     vth(str,b)*y(str,b))/ ((y(str,b)+y(str,c))*solrot * vbp)

      write(6,500)'omega rb = ',wb
      write(6,500)'omega rc = ',wc
```

```
      write(6,500)'Mrot b = ',wb/a(str,b)
      write(6,500)'Mrot c = ',wc/a(str,c)
      write(6,*)
      write(6,500)'Vb = ',vb
      write(6,500)'Vb‘ = ',vbp
      write(6,500)'Vc‘ = ',vcp
      write(6,500)'Vc = ',vc
      write(6,*)
      write(6,500)'Mb = ',vb / a(str,b)
      write(6,500)'Mb‘ = ',vbp / a(str,b)
      write(6,500)'Mc‘ = ',vcp / a(str,c)
      write(6,500)'Mc = ',vc / a(str,c)
      write(6,*)
      write(6,500)'Beta b‘ = ',-180.0/3.14159*atan(th(6)/me(6))
      write(6,500)'Beta c‘ = ',-180.0/3.14159*atan(th(12)/me(12))
      write(6,500)'Beta c = ',-180.0/3.14159*atan(th(8)/me(8))
      write(6,*)
      write(6,500)'r in:  ',y(str,b)
      write(6,500)'r out:  ',y(str,c)
      write(6,*)
      write(6,500)'Solidity in rotor:  ',solrot
      write(6,500)'D in rotor:  ',drot

      call grklin(ilin,isym,nper,titl,6,th,me,
     &    '~m/s~m/s~Rotor Velocity Triangle',23)

      write(6,*)

C Make Stator velocity triangle

C Vc
      nper(1) = 2
      ilin(1) = 4
      isym(1) = 0
      titl(1) = 'Vc         '
      th(1) = 0.0
      me(1) = 0.0
      th(2) = -1.0*vth(str,c2)
      me(2) = vm(str,c2)
      vc = sqrt(th(2)**2+me(2)**2)

C Vd
      nper(2) = 2
      ilin(2) = 1
      isym(2) = 7
      titl(2) = 'Vd         '
      th(3) = 0.0
      me(3) = 0.0
      th(4) = vth(str,d)
      me(4) = vm(str,d)
      vd = sqrt(th(4)**2+me(4)**2)

      solstat = sch / (2*3.14159*y(str,d)/nstat)
      dstat = 1- vd/vc + (abs(vth(str,d)*y(str,d)-vth(str,c)*
```

```
     &     y(str,c)))/((y(str,c)+y(str,d))*solstat*vc)

      write(6,500)'Vc = ',vc
      write(6,500)'Vd = ',vd
      write(6,*)
      write(6,500)'Mc = ',vc/a(str,c2)
      write(6,500)'Md = ',vd/a(str,d)
      write(6,*)
      write(6,500)'Beta c = ',-180.0/3.14159*atan(th(2)/me(2))
      write(6,*)
      write(6,500)'r in:  ',y(str,c2)
      write(6,500)'r out:  ',y(str,d)
      write(6,*)
      write(6,500)'Solidity in stator:  ',solstat
      write(6,500)'D in stator:  ',dstat

      call grklin(ilin,isym,nper,titl,2,th,me,
     &     '~m/s~m/s~Stator Velocity Triangle',23)

      return
      end

C-----
```

MISES interface - savecam.f

```
C Write the camber lines for a rotor or stator streamline

      subroutine savecam
      include 'vars.inc'
      include 'fns.inc'

      integer str,b,c,rsv,sp,sm
      integer lpt,npts,nbl
      real*8 z1,z3,bp,bpo,zpi,zpo,zpc,tpc,tpi
      real*8 ch,chtip,gang,gangtip,reyn,ir,ypt,ppa
      real*8 zp(500),tp(500),rp(500)
      real*8 mpu(100),thu(100),mpl(100),thl(100)
      real*8 zrp(500),trp(500),zsta,zend,inum
      real*8 x0,y0,rad,ang0,ang1,dx,da
      integer sx0,sx1,st0,st1
      real*8 strcomp(256,3),offset,sfl,angin,angout
      real*8 thick,hc,disp,theta,sinl,sout,ang,mch,rot,mchout
      real*8 tipthick,xc,yc,sol,dfac,xcen,ycen,xmov,ymov
      real*8 rch,sch,b1,b2,xi,yi,sta,spa
      real*8 dinl,ainl,amin,angt,angb
      real*8 minl,aast,spc,h1,h2,xt
      integer oflag
```

```
      character*4 fsu,check
      character*10 fsn
      character*11 fsn2
      character*9 fsn3
      character*16 cname

 500  format(a,f7.3)
 505  format(a4)

      oflag = 1
      if (oflag.eq.1) then
        write(6,*) 'Regular grid selected in code'
      else if (oflag.eq.-1) then
        write(6,*) 'Offset periodic grid selected in code'
      else
        write(6,*) 'oflag set incorrectly.   Check SC code.'
        goto 3000
      end if
      write(6,*)

 50   write(6,*) 'Enter file suffix ID for MISES (4 char max)'
      write(6,*) 'Or enter xxxx to write out all blade info to a file'

      read(5,505) fsu

      check = 'xxxx'

      if (fsu(1:4).eq.check(1:4)) goto 2000

      str = 10*(ichar(fsu(2:2))-48)+ichar(fsu(3:3))-48

      write(6,*) 'Using streamline:  ',str

      rsv = 0
      if (fsu(1:1).eq.'r') then
        rsv = 1
        write(6,*) 'Rotor Streamline'
      else if (fsu(1:1).eq.'s') then
        rsv = 2
        write(6,*) 'Stator Streamline'
      end if
      if (rsv.eq.0) then
        write(6,*) 'Enter 1 for a rotor, 2 for a stator'
        read(5,*) rsv
      end if

      goto (101,102) (rsv)

      write(6,*)'Try again!'
      goto 50

 101  b = rosta
      c = roend
      nbl = nrot
```

```
      rot = omega
      thick = (rhubtc+(rtiptc-rhubtc)*
     &     w(y(1,rosta),y(str,rosta),y(numr,rosta)))
      mch = mtorel(str,b)
      mchout = sqrt((omega*y(str,c)-vth(str,c))**2+(vm(str,c))**2)/
     &     a(str,c)
      angin = -1.0*betarel(str,b)
      angout = -1.0*betarel(str,c)
      minl = mch

      goto 200

 102  b =ststa
      c = stend
      nbl = nstat
      thick = (shubtc+(stiptc-shubtc)*
     &     w(y(1,ststa),y(str,ststa),y(numr,ststa)))
      mch = mto(str,b)
      mchout = mto(str,c)
      rot = 0.0
      angin = beta(str,b)
      angout = beta(str,c)
      minl = mch

C find the points

 200  continue

      sinl = tan(angin)
      sout = tan(angout)

      zsta = z(str,b)
      zend = z(str,c)
      dx = sqrt((y(str,c)-y(str,b))**2 + (zend-zsta)**2)/y(str,c)*1.2

C  add some degrees to inlet angle to make flow better (an estimate)
      if (rsv.eq.1) then
        ang0 = angin + 2.0*3.14159/180.0
      else
        ang0 = angin + 0.5*3.14159/180.0
      end if
C  Add fraction of turning to outlet angle for deviation (an estimate)
      if (rsv.eq.1) then
        ang1 = angout - .27*(angin-angout)
      else
        ang1 = angout - .38*(angin-angout)
      end if
C Estimate chord and spacing
      ch = dx/cos((angin+angout)/2.0)
      spa = 2*3.14159*y(str,b)/nbl

C Compute the upper surface rounded nose
      disp = ch/150.0
      ang = ang0
```

```
    xc = disp*cos(ang)
    yc = disp*sin(ang)

    do i = 1,9
      ir = ((1.0-i*1.0)/16.0+1.0)*3.14159+ang
      mpu(i) = xc+disp*cos(ir)
      thu(i) = yc+disp*sin(ir)
    end do

C Compute the entry arc
    if (str.eq.1) then
      h1 = y(2,b)-y(1,b)
      h2 = y(2,c)-y(1,c)
    else if (str.eq.numr) then
      h1 = y(str,b)-y(str-1,b)
      h2 = y(str,c)-y(str-1,c)
    else
      h1 = y(str+1,b)-y(str-1,b)
      h2 = y(str+1,c)-y(str-1,c)
    end if

    xt = spa/2.0*sin(2.0*ang0)
    aast = 1.0/minl*((1.0+0.2*minl**2.0)/1.2)**3.0

    ht = h1+(h2-h1)*w(0,xt,dx)

    dinl = spa*cos(ang0)

    ainl = dinl*h1
    amin = ainl/aast

    if (rsv.eq.1) then
      angt = acos(amin/(ht*spa))+(ang1-ang0)/8.0
    else
      angt = acos(amin/(ht*spa)) - ang0/8.0
    end if

C Make an arc from ang0 to angt
C Find circular arc blade shape given
C the slopes of the inlet and outlet and the streamwise distance
C The center of the circle is at (0,0)

    da = spa/2.0*sin((ang0+angt))
    x0 = -1.0*da*sin(ang0)/(sin(ang0)-sin(angt))
    x1 = x0+da
    y0 = -1.0/tan(ang0)*x0
    y1 = -1.0/tan(angt)*x1
    rad = sqrt(x0**2+y0**2)
C Move arc to correct position
    xmov = mpu(9)-x0
    ymov = thu(9)-y0

    xcen = 0.0+xmov
    ycen = 0.0+ymov
```

```
      x1 = x1+xmov
      y1 = y1+ymov
      x0 = x0+xmov
      y0 = y0+ymov

      do i = 10,29
         mpu(i} = mpu(9)+(i-9.0)/20.0*(x1-mpu(9))
         thu(i) = ycen+sqrt(rad**2-(mpu(i)-xcen)**2)
      end do

C Compute the rear circular arc region
C Make an arc from angt to angl

      da = dx-da
      x0 = -1.0*da*sin(angt)/(sin(angt)-sin(angl))
      x1 = x0+da
      y0 = -1.0/tan(angt)*x0
      y1 = -1.0/tan(angt)*x1
      rad = sqrt(x0**2+y0**2)

C Move beginning of arc to end of entry region
      xmov = mpu(29)-x0
      ymov = thu(29)-y0

      xcen = 0.0+xmov
      ycen = 0.0+ymov
      x1 = x1+xmov
      y1 = y1+ymov
      x0 = x0+xmov
      y0 = y0+ymov

C Compute the real chord
      ch = sqrt(x1**2+y1**2)

      do i = 30,49
         mpu(i) = mpu(29)+(i-29.0)/20.0*(x1-mpu(29))
         thu(i) = ycen+sqrt(rad**2-(mpu(i)-xcen)**2)
      end do

C     Compute the rounded nose for the bottom surface

      do i = 1,8
         ir = ((i*1.0)/16.0+1.0)*3.14159+ang
         mpl(i) = xc+disp*cos(ir)
         thl(i) = yc+disp*sin(ir)
      end do

C Make an arc on the bottom surface with a entrance slope equal to
C the upper surface slope

      x0 = mpl(8)
      y0 = thl(8)

C Add trailing edge thickness
```

```
      x1 = mpu(49)+disp/2.0*sin(ang1)
      y1 = thu(49)-disp/2.0*cos(ang1)

C Add a wedge angle to the lower surface
      if (rsv.eq.2) then
        angb = ang0-0.0/180*3.14159
      else
        angb = ang0-7.0/180*3.14159
      end if

      xcen = (x1**2-x0**2+(y1-y0)**2-2.0*(y1-y0)*x0/tan(angb))/
     &     (2.0*(x1-x0)-2.0*(y1-y0)/tan(angb))
      ycen = y0-(xcen-x0)/tan(angb)

      rad = sqrt((x0-xcen)**2+(y0-ycen)**2)

      sta = atan((y1-thl(8))/(x1-mpl(8)))

      do i = 9,49
        mpl(i) = mpl(8)+(i-8.0)/41.0*(x1-mpl(8))
        thl(i) = ycen+sqrt(rad**2-(mpl(i)-xcen)**2)
      end do

C Write out the 'blade.xxx' file
 21   format(f10.5,f10.5,f10.5,f10.5,f10.5)
 22   format(f12.7,f12.7)
 38   format(a32)

      sol = ch/(2*3.14159*y(str,c)/nbl)
      dfac = 1 - mchout/mch + abs(mth(str,c)*y(str,c)-
     &     mth(str,b)*y(str,b))/ ((y(str,b)+y(str,c))*sol*mch)

      fsn(1:6) = 'blade.'
      fsn(7:10) = fsu
      open(unit=1,file=fsn,status='UNKNOWN')

      write(6,*)'Enter 32 character max name of case:'
C       read(5,38) cname

      if (rsv.eq.1) then
        cname(1:14) = 'ROTOR STREAM  '
      else
        cname(1:14) = 'STATOR STREAM '
      end if

      cname(15:16) = fsu(2:3)
      write(6,38) cname
      write(1,38) cname
      write(1,21) sinl,sout,0.2,0.2,2*3.14159/nbl

C Write top surface
      do i = 49,1,-1
        write(1,22) mpu(i),thu(i)
      end do
```

```
C Write bottom surface
      do i = 1,49
        write(1,22) mpl(i),thl(i)
      end do

      close(1)

C make 'stream.xxx' file
      fsn2(1:7) = 'stream.'
      fsn2(8:11) = fsu

      open(unit=1,file=fsn2,status='UNKNOWN')

      write(1,*) -1.0*oflag*rot*ch/sqrt(g*r*tt(str,1))

      sp = str+1
      sm = str-1

      if (str.eq.1) then
        sm = 1
      else if (str.eq.numr) then
        sp = numr
      end if

      offset = 0

      do i = b,c
        strcomp(i-b+1,1) = sqrt((z(str,i)-
     &        z(str,b))**2+(y(str,i)-y(str,b))**2)/y(str,i)
        strcomp(i-b+1,2) = y(str,i)/ch
        strcomp(i-b+1,3) = (y(sp,i)-y(sm,i))/ch
      end do

24    format(f11.5,f11.5,f11.5)
      do i = 10,1,-1
        ir = i*1.0/10.0
        write(1,24) strcomp(1,1)-offset-2.0*ir,
     &        strcomp(1,2),strcomp(1,3)
      end do

      do i = 1,c-b+1
        write(1,24) strcomp(i,1)-offset,strcomp(i,2),strcomp(i,3)
      end do

      do i = 1,10
        ir = i*1.0/10.0
        write(1,24) strcomp(c-b+1,1)-offset+2.0*ir,
     &        strcomp(c-b+1,2),strcomp(c-b+1,3)
      end do

      close(1)

      write(6,*) 'Files ',fsn,' and ',fsn2,' saved.'
```

```
C Write an 'ises.xxx' file.  This may need to be changed depending on
C the boundary conditions and design/analysis mode.

 31  format(f7.3,f7.3,f7.3,f7.3,f6.2,a)
 32  format(e11.3,f6.2,f7.3,f7.3,a)
     fsn3(1:5) = 'ises.'
     fsn3(6:9) = fsu
     open(unit=1,file=fsn3,status='UNKNOWN')
C      if (rsv.eq.2) then
C        write(1,*) '1,2,5,,,,,,,,,,,,,,,,,,,,,,,'
C        write(1,*) '1,3,4,,,,,,,,,,,,,,,,,,,,,,,'
C      else
        write(1,*) '1,2,5,15,,,,,,,,,,,,,,,,,,,,,'
        write(1,*) '16,3,4,18,,,,,,,,,,,,,,,,,,,,,'
C      end if
     if (rsv.eq.2) then
        ppa = (1.0+(g-1.0)/2.0*mto(str,c)**2)**(-3.5)*pt(str,c)/
    &        pt(str,b)
     else
        ppa = p(str,c)/pt(str,b)
     end if

     write(1,31) mch,sinl*oflag,sout*oflag,ppa,0.0,
    &     '   | MACH SINL SOUT P2/P0a MFRIN'
     reyn = rho(str,b)*vto(str,b)*ch/1.86e-5
     write(1,32) 0.0,6.0,0.02,0.02,'       | RE ACRIT XTRS XTRP'
     write(1,*) '3    0.2    0.90    1.0                    | ISMOM PCWT'
    &     ,' MCRIT MUCON'
     write(1,*) '0   0                                      | NITER IGLOSEN'
     write(1,*) '0.0 0.0 0.  0.  0.  0.  0.  0.  0.  0.  0.  | Dmov Drot ',
    &     ' Dmod1-9'
     write(1,*)

 100  format(a,f12.6,a,f12.6)
 104  format(a,f12.6,a,e12.6)
     write(1,100)'MACHin = ',mch,' MACHout = ',mchout
     write(1,*)
     write(1,100)'ANGIN = ',angin*180/3.14159,
    &     ' ANGOUT = ',angout*180/3.14159
     write(1,*)
     write(1,104)'D = ',dfac,' REYN = ',reyn
     write(1,*)
     write(1,100)'p2/p0a = ',ppa

     close(1)

     write(6,*) fsn3,' has been saved.    This may need to be changed'
     write(6,*) 'depending on boundary conditions and design mode.'
     goto 3000

 2000 continue
C Write out general info

     open(unit=1,file='bladinfo',status='UNKNOWN')
```

```
 2059 format(i7,f10.3,f8.2,f8.2,f7.3,f7.3,e11.3)

C Start writing rotor info

      b = rosta
      c = roend
      nbl = nrot
      thick = rthick
      rot = omega

      write(1,*) 'ROTOR STREAMLINE INFO'
      write(1,*) 'stream     mp-ch    angin   angout     Min     Mout      Re
     &        rin    rout'

      do str = 1,numr
        mch = sqrt(vm(str,b)**2+(omega*y(str,b))**2)/a(str,b)
        mchout = sqrt((omega*y(str,c)-vth(str,c))**2+(vm(str,c))**2)/
     &        a(str,c)

        angin = 180.0/3.14159*atan(-1.0*(vth(str,b)-rot*y(str,b))/
     &        vm(str,b))
        angout = 180.0/3.14159*atan(-1.0*(vth(str.c)-rot*y(str,c))/
     &        vm(str,c))

        zsta = z(str,b)
        zend = z(str,c)
        if (y(str,c).eq.y(str,b)) then
          ch = (zend-zsta)/y(str,c)
        else
          ch = sqrt(((zend-zsta)/(y(str,c)-y(str,b)))**2+1.0)*
     &          log(y(str,c)/y(str,b))
        end if
        ch = abs(ch)
        reyn = rho(str,b)*vto(str,b)*ch/1.86e-5

        write(1,2059) str,ch,angin,angout,mch,mchout,reyn
      end do

C Write stator info
      b = ststa
      c = stend
      nbl = nstat
      thick = sthick
      rot = 0.0

      write(1,*)
      write(1,*) 'STATOR STREAMLINE INFO'
      write(1,*) 'stream     mp-ch    angin   angout     Min     Mout      Re
     &        rin    rout'

      do str = 1,numr
        mch = mto(str,b)
        mchout = mto(str,c)
```

```
     angin = 180/3.14159*atan(-1.0*(vth(str,b)-rot*y(str,b))/
  &        vm(str,b))
     angout = 180/3.14159*atan(-1.0*(vth(str,c)-rot*y(str,c))/
  &        vm(str,c))

     zsta = z(str,b)
     zend = z(str,c)
     if (y(str,c).eq.y(str,b)) then
        ch = (zend-zsta)/y(str,c)
     else
        ch = sqrt(((zend-zsta)/(y(str,c)-y(str,b)))**2+1.0)*
  &          log(y(str,c)/y(str,b))
     end if
     ch = abs(ch)
     reyn = rho(str,b)*vto(str,b)*ch/1.86e-5
     write(1,2059) str,ch,angin,angout,mch,mchout,reyn
   end do

 3000 return

   end
```

Compute radius of curvature - roc.f

```
   subroutine roc

C This fills xr,yr,rds in the common block for a given
C x1,x2,x3,y1,y2,y3

C Three points determine a circle, and the center is at the
C intersection of the perpendicular bisectors

   include 'vars.inc'

   real*8 sa,sb,xa,xb,ya,yb

   xa = (x1+x2)/2
   ya = (y1+y2)/2
   sa = (x1-x2)/(y2-y1)

   xb = (x3+x2)/2
   yb = (y3+y2)/2
   sb = (x3-x2)/(y2-y3)

   if (y2.eq.y3) then
     xr = xb
     yr = ya+sa*(xr-xa)
   else if (y2.eq.y1) then
     xr = xa
     yr = yb+sb*(xr-xb)
   else
     xr = (sb*xb-yb-sa*xa+ya)/(sb-sa)
```

```
        yr = ya+sa*(xr-xa)
      end if

      rds = sqrt((x1-xr)**2+(y1-yr)**2)

      if (yr.gt.y2) rds = -1.0*rds

      return

      end
```

Grid generation program - gridfan.f

```
C procedure for defining initial streams and geometry
C The top stream y(numr,m) must have a larger y coordinate
C than the bottom stream y(1,m)

C This program gridfan.f is separate

      implicit none
      integer maxm,maxr
      parameter(maxr=33,maxm=256)

      dimension yy(maxm,maxr),zz(maxm,maxr)
      real yy,zz
      character*12 grname,dfname
      character*4 id

      integer i,j,plottype,numm,numr
      integer rosta,roend,ststa,stend,exitm
      real ytop,ybot,ybot1,ybot2,ybot3,ybot4,ybot5,ybot6
      real mdotstr,psrat,dx,mcru,patm,tatm
      real w,ttf,omega,minlet,visc

      write(6,*)'Enter a four character id name for the run'
      read(5,1000) id
 1000 format(a4)
      grname(1:4) = id
      grname(5:12) = 'grid.dat'
      dfname(1:4) = id
      dfname(5:12) = 'data.dat'


      write(6,*)'Enter the number of points in the r-direction'
      write(6,*)'This must be an odd number'
      write(6,*)'The maximum is:  ',maxr
      read(5,*) numr

      write(6,*)'Point m=1 is the duct entrance'

      write(6,*)'Enter the m value of the rotor start'
      write(6,*)'Be sure that all m values are increasing integers'
      read(5,*) rosta
```

```
write(6,*)'Enter the m value of the rotor end'
read(5,*) roend

write(6,*)'Enter the m value of the stator start'
read(5,*)ststa

write(6,*)'Enter the m value of the stator end'
read(5,*) stend

write(6,*)'Enter the m value of the duct exit'
read(5,*) exitm
numm = exitm

write(6,*)'Enter the y value of the top of the duct'
read(5,*) ytop

write(6,*)'Enter the m=1 y bottom coordinate.'
read(5,*) ybot1

write(6,*)'Enter the y value of the bottom of the rotor start'
read(5,*) ybot2

write(6,*)'Enter the y value of the bottom of the rotor end'
read(5,*) ybot3

if (ststa .ne. roend) then
  write(6,*)'Enter the y value of the bottom of the stator start'
  read(5,*) ybot4
else
  ybot4 = ybot3
end if

write(6,*)'Enter the y value of the bottom of the stator end'
read(5,*) ybot5

write(6,*)'Enter the y value of the duct exit'
read(5,*) ybot6

ttf = 1.27

write(6,*)'Enter duct inlet mach number'
read(5,*) minlet

write(6,*)'Enter wheel rotation freq.  (rad/s)'
read(5,*) omega

write(6,*)'Enter the grid spacing in x (m)'
read(5,*) dx

patm = 37000.0
tatm = 222.0
mcru = 0.8
```

```
      plottype = 1

      do i = 1.numm

        do j = 1,numr
          zz(i,j) = (i-rosta)*dx
        end do

        if (i.lt.rosta) then
          ybot = ybot1+w(1,i,rosta)*(ybot2-ybot1)
        else if (i.lt.roend) then
          ybot = ybot2+w(rosta,i,roend)*(ybot3-ybot2)
        else if (i.lt.ststa) then
          ybot = ybot3+w(roend,i,ststa)*(ybot4-ybot3)
        else if (i.lt.stend) then
          ybot = ybot4+w(ststa,i,stend)*(ybot5-ybot4)
        else
          ybot = ybot5+w(stend,i,exitm)*(ybot6-ybot5)
        end if

        yy(i,1) = ybot
        yy(i,numr) = ytop

        mdotstr = 0.5*(yy(i,numr)**2-yy(i,1)**2)/(numr-1.0)

        do j = 2,numr-1
          yy(i,j) = sqrt(yy(i,j-1)**2+2*mdotstr)
        end do
      end do

C Write out grid and datafile

      open(file=grname,unit=1)
      do i = 1,numr
        do j = 1,numm
          write(1,*) zz(j,i),yy(j,i)
        end do
      end do
      close(1)

      write(6,*) grname,' is made.'

      open(2,file=dfname)

 11   format(i30,a)
 12   format(f30.5,a)
 21   format(i15,i15,a)
 22   format(f15.5,f15.5,a)

      write(2,21) numr,numm,'    !  numr,numm'
      write(2,21) rosta,roend,'    !  rotor start, end'
      write(2,21) ststa,stend,'    !  stator start, end'
      write(2,22) 1.23,minlet,'    !  Fan Temp ratio, Inlet Mach no.'
      write(2,12) omega,'    !  rotation frequency'
```

```
      write(2,22) patm,tatm,'   !  Atm.  Press., Temp.'
      write(2,12) mcru,'    !  Cruise mach number'
      write(2,22) 0.05,0.05,'   !  Rot loss, Stat loss'
      write(2,22) 0.08,0.04,'   !  Rot hub t/c, tip t/c'
      write(2,22) 0.08,0.04,'   !  Stat hub t/c, tip t/c'
      write(2,21) 32,49,'   !  num rotor blades,num stator blades'
      write(2,11) plottype,'   !  Plottype'
      close(2)

      write(6,*) dfname,' is made.'

      end

C--------------------

      real function w(p1,p2,p3)

      real p1,p2,p3

      if (p2.lt.p1) then
        w = 0.0
      else if (p2.gt.p3) then
        w = 1.0
      else
        w = (p2-p1)/(p3-p1)
      end if

      return
      end
```

Appendix B

Velocity Triangles

Figures B-1 through B-10 show the velocity triangles for the streamlines computed by SC, the streamline curvature throughflow code.

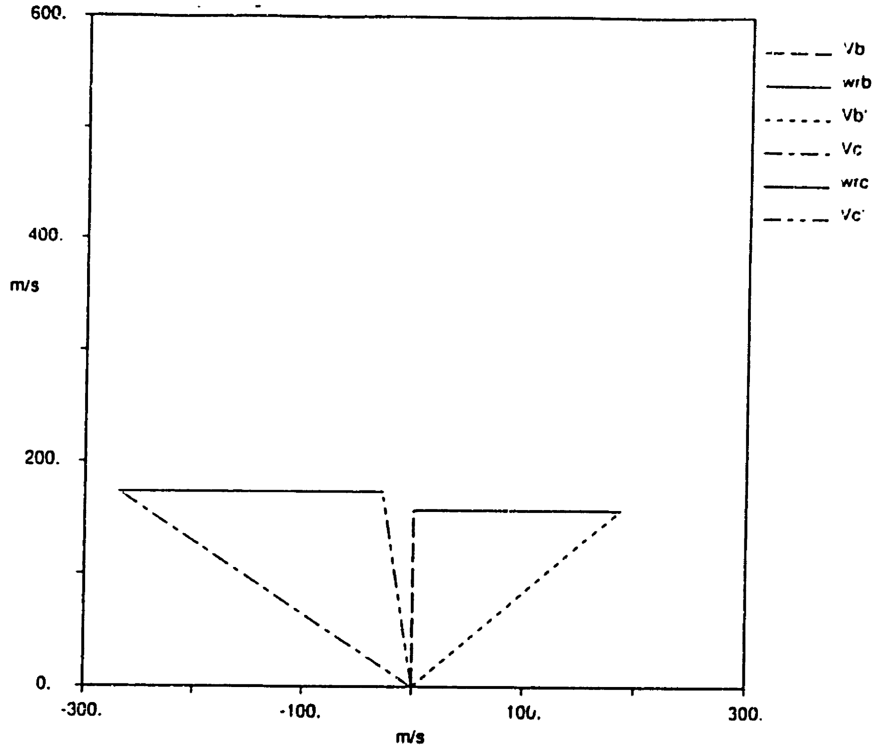

Figure B-1: Rotor hub velocity triangle

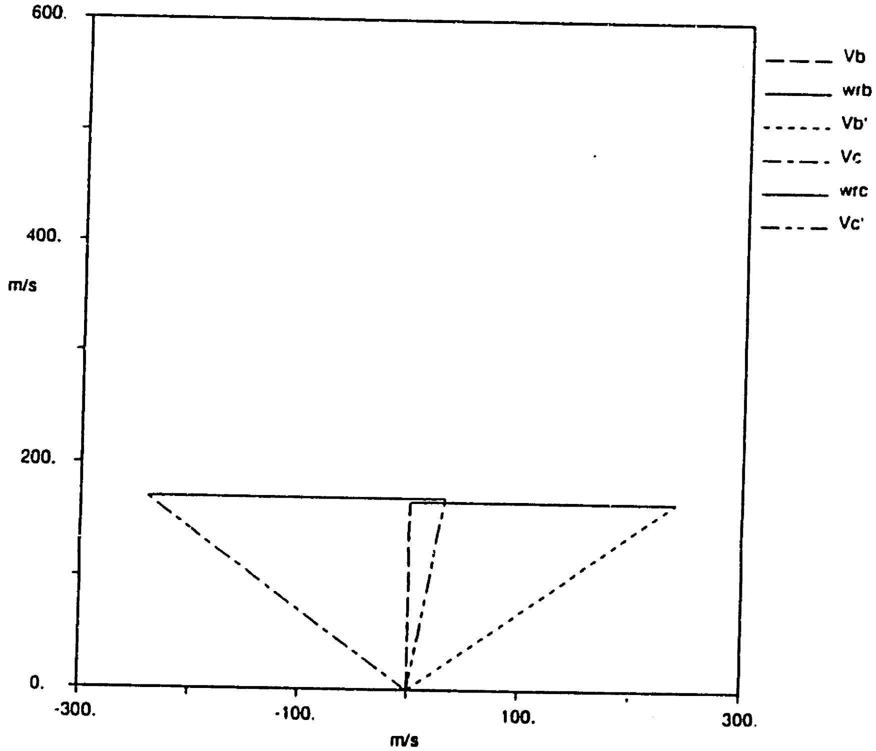

Figure B-2: Rotor 1/4 span velocity triangle

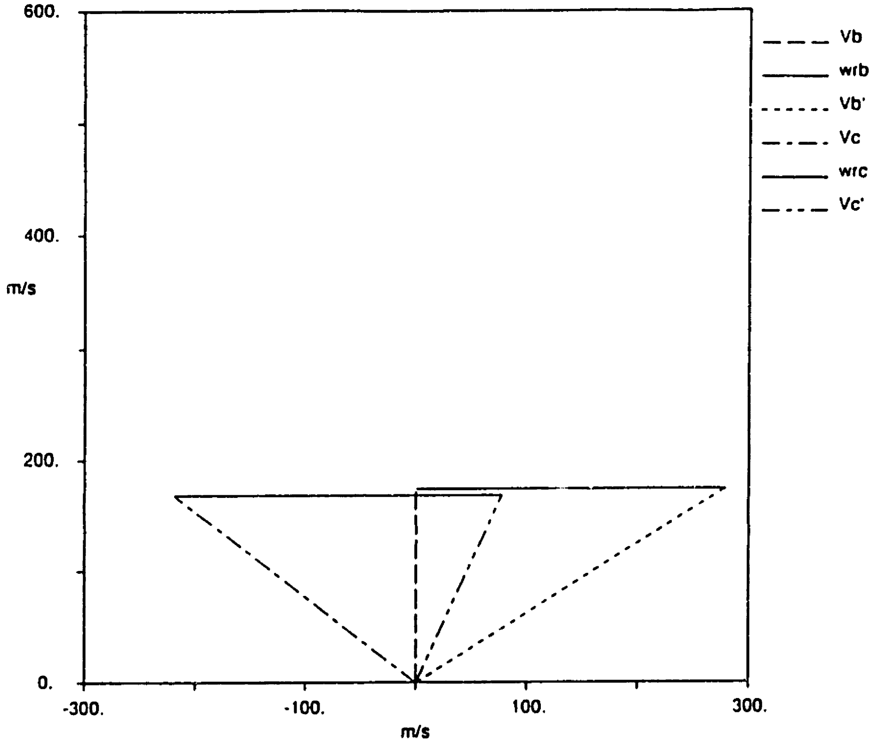

Figure B-3: Rotor 1/2 span velocity triangle

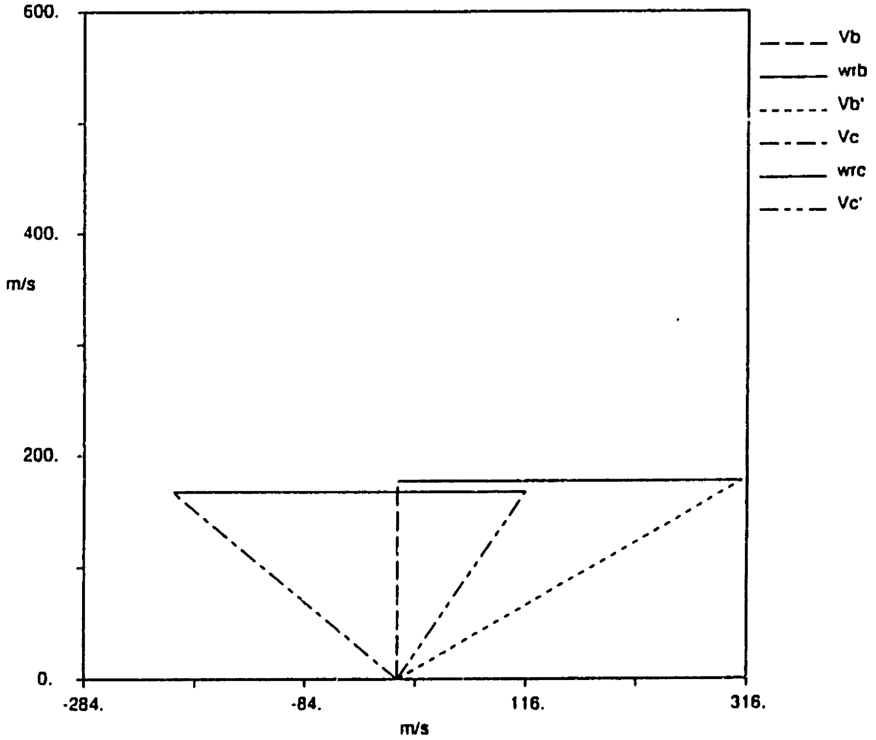

Figure B-4: Rotor 3/4 span velocity triangle

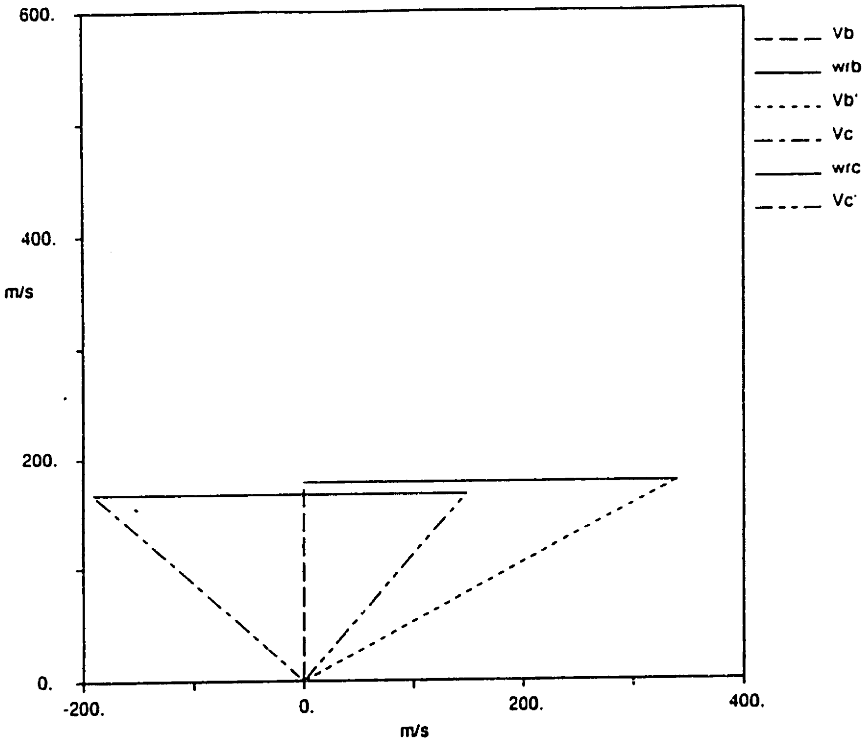

Figure B-5: Rotor tip velocity triangle

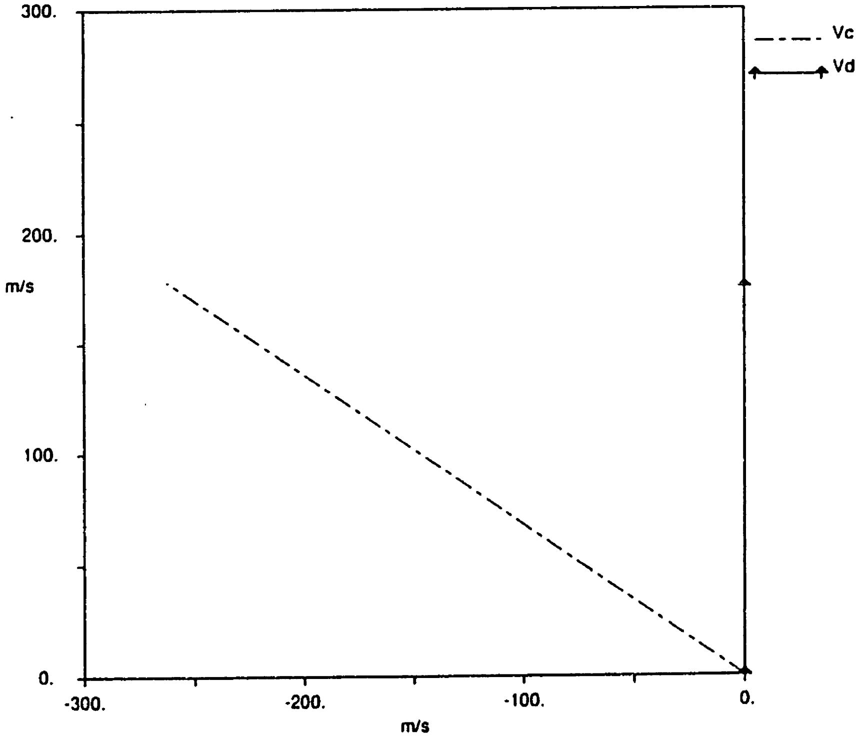

Figure B-6: Stator hub velocity triangle

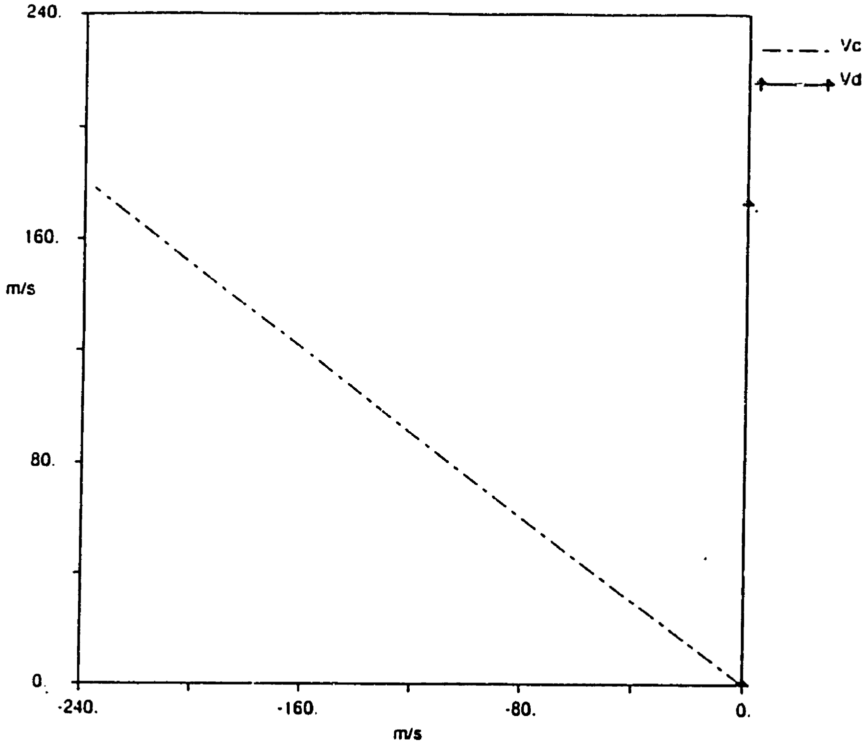

Figure B-7: Stator 1/4 span velocity triangle

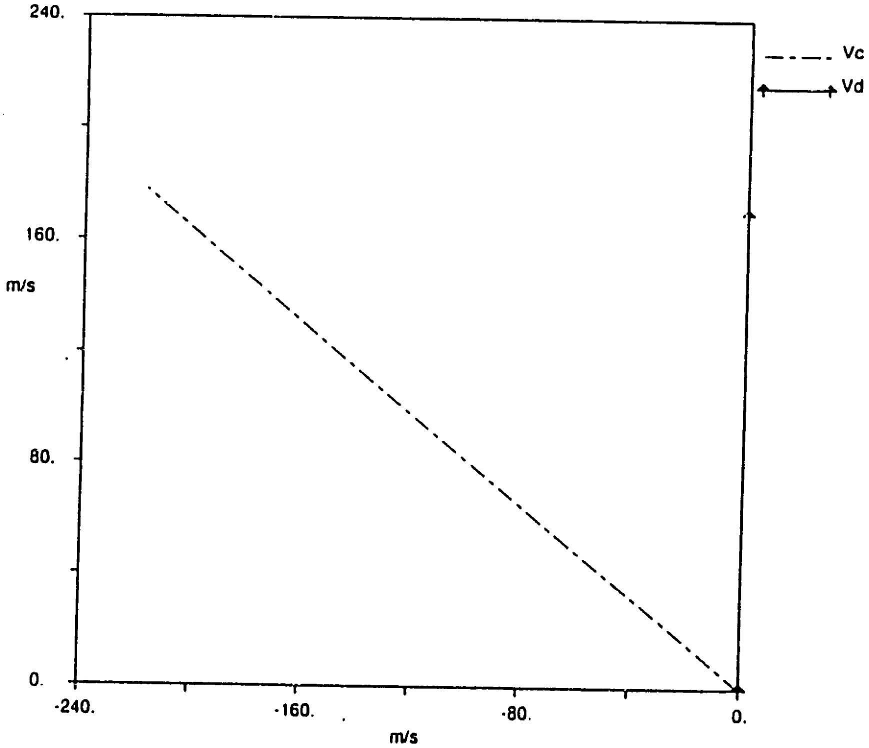

Figure B-8: Stator 1/2 span velocity triangle

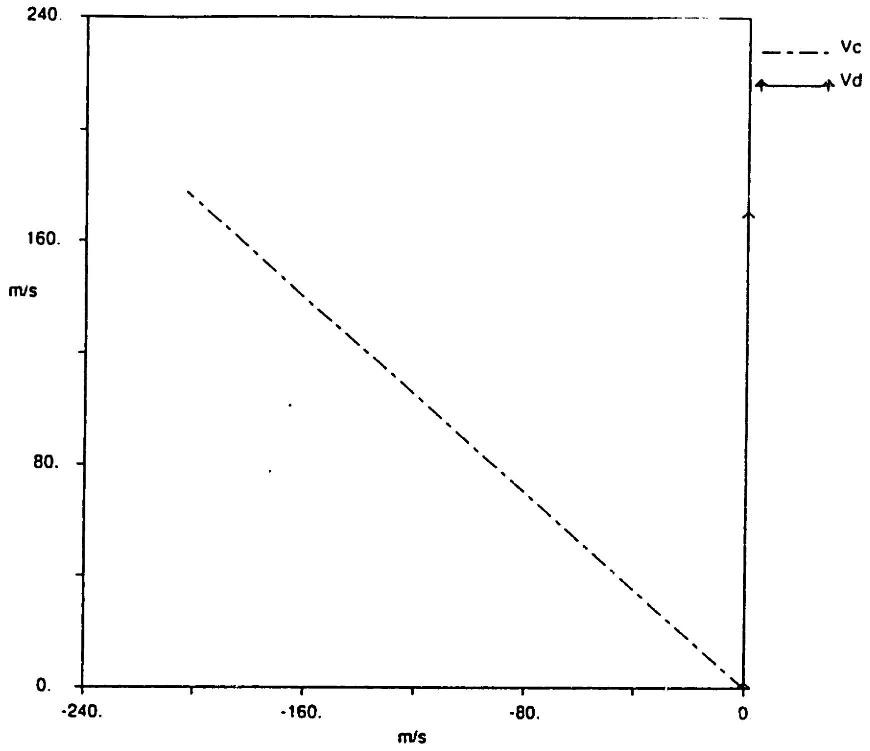

Figure B-9: Stator 3/4 span velocity triangle

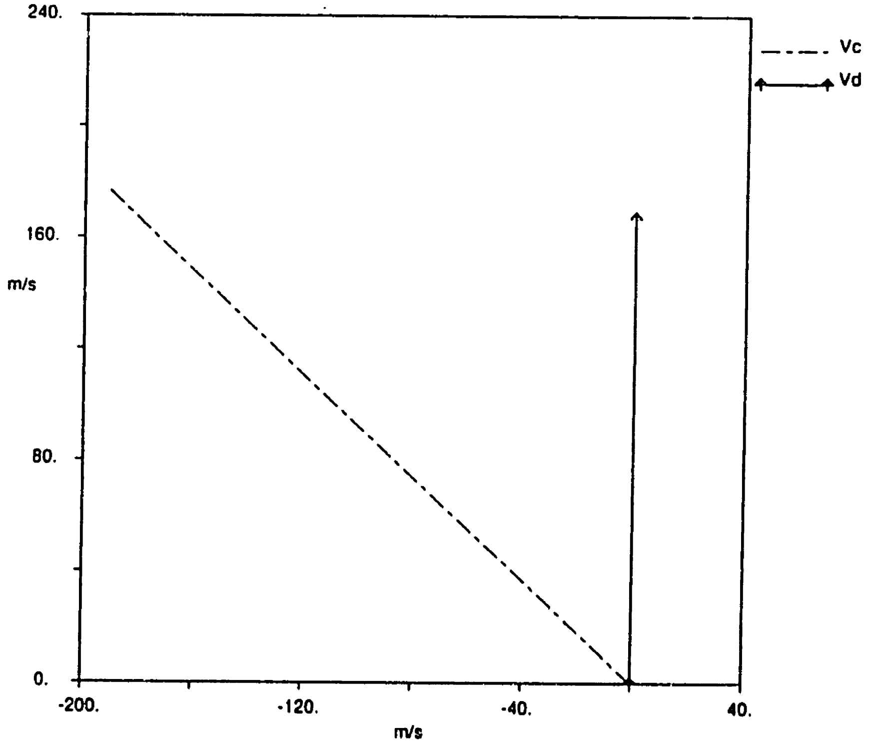

Figure B-10: Stator tip velocity triangle

Appendix C

Blade Sections

For all following plots, each contour line is .1 Mach. The boundary layer thicknesses are nondimensionalized by the nondimensional radius at that station, which is the actual radius divided by the blade chord.

Figures C-1 through C-4 show the best solution found for the rotor hub. The solution was not converged.

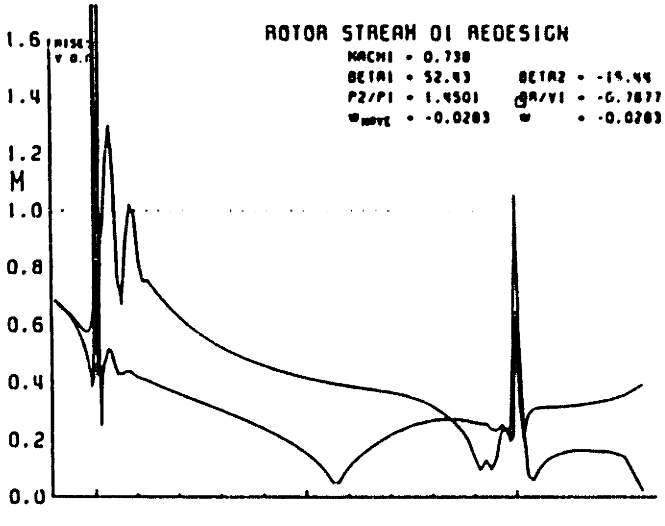

Figure C-1: Surface Mach distribution - Rotor hub

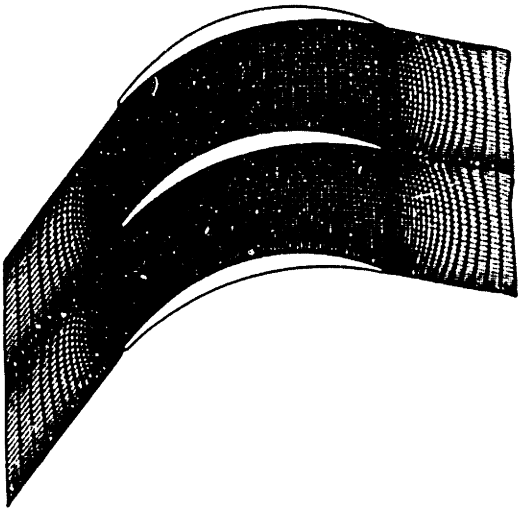

Figure C-2: Computation grid - Rotor hub

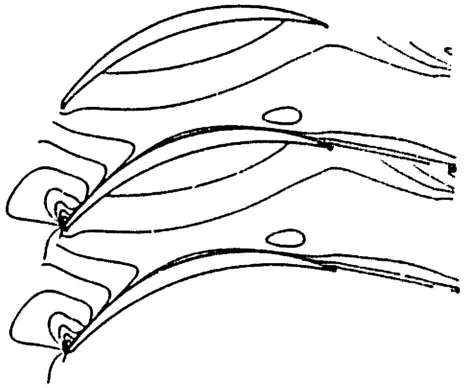

Figure C-3: Contour Mach plot - Rotor hub

NOT COMPUTED

Figure C-4: Suction side boundary layer thickness - Rotor hub

Figures C-5 through C-8 show the solution found for the rotor 1/4 span.

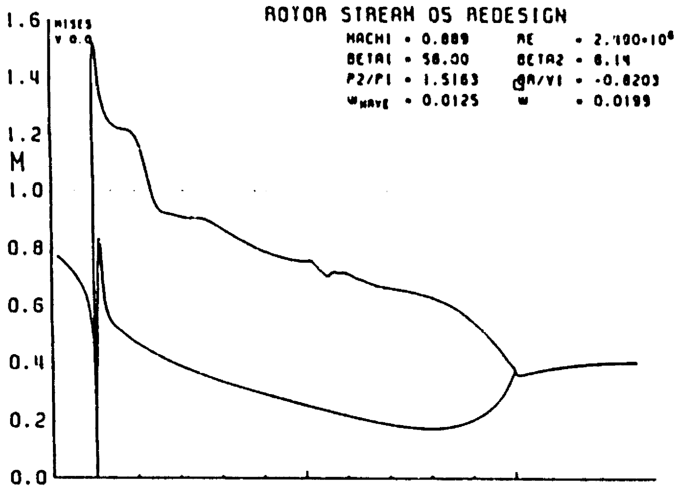

Figure C-5: Surface Mach distribution - Rotor 1/4 span

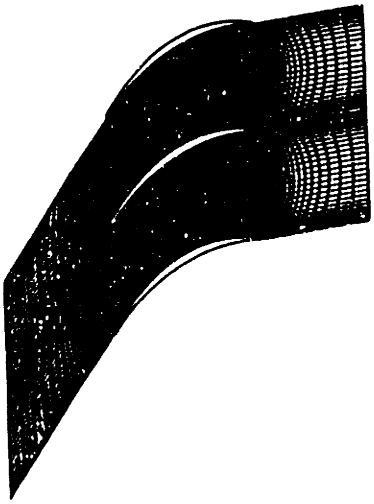

Figure C-6: Computation grid - Rotor 1/4 span

Figure C-7: Contour Mach plot - Rotor 1/4 span

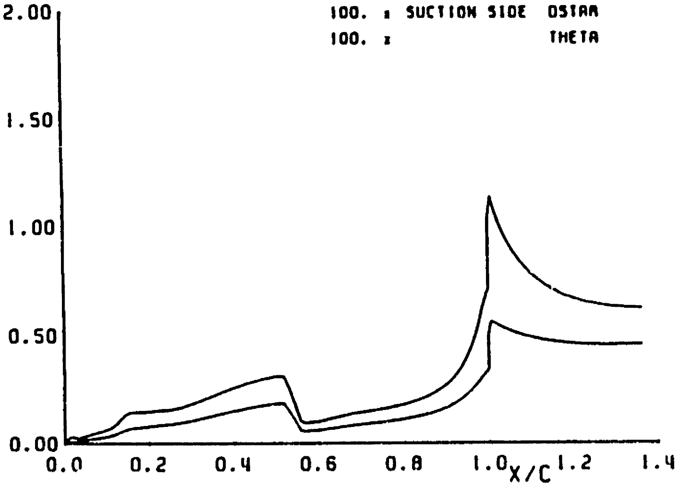

Figure C-8: Suction side boundary layer thickness - Rotor 1/4 span

Figures C-9 through C-12 show the solution found for the rotor 1/2 span.

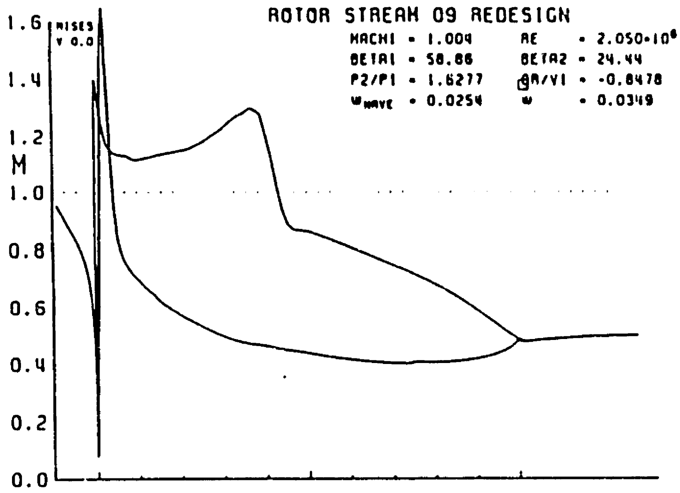

Figure C-9: Surface Mach distribution - Rotor 1/2 span

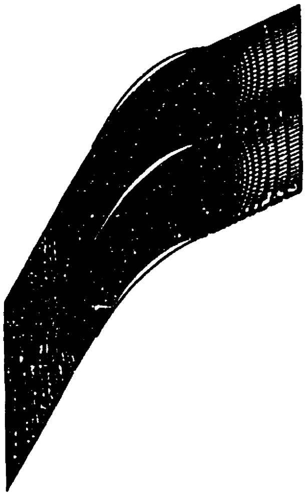

Figure C-10: Computation grid - Rotor 1/2 span

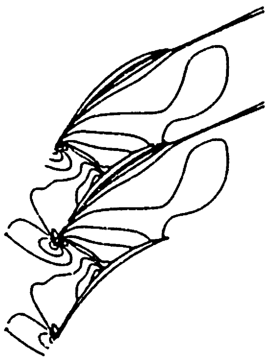

Figure C-11: Contour Mach plot - Rotor 1/2 span

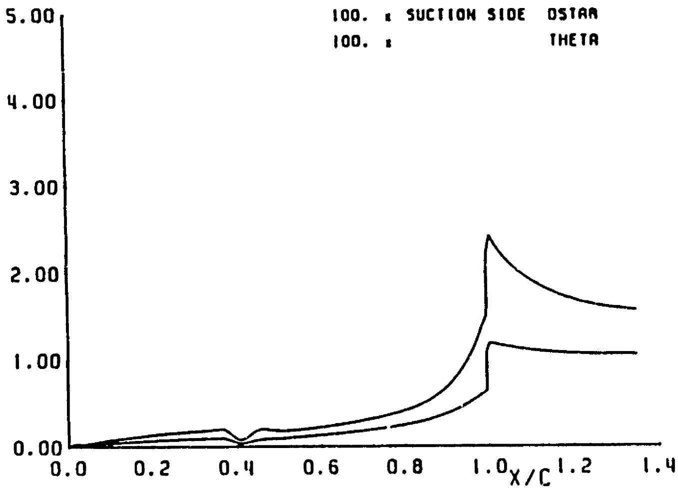

Figure C-12: Suction side boundary layer thickness - Rotor 1/2 span

Figures C-13 through C-16 show the solution found for the rotor 3/4 span.

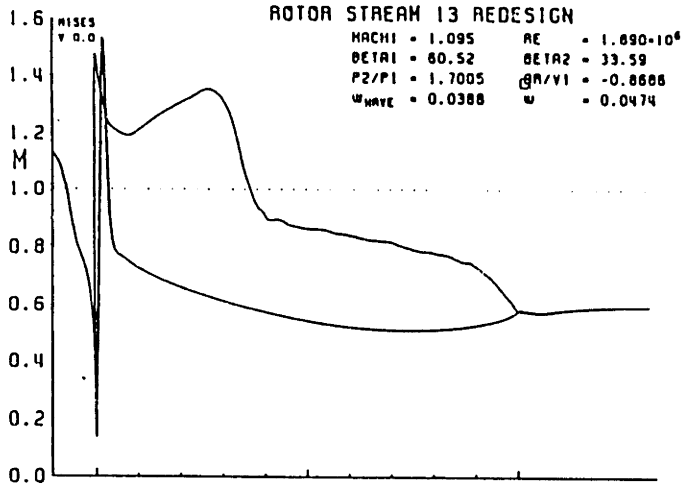

Figure C-13: Surface Mach distribution - Rotor 3/4 span

Figure C-14: Computation grid - Rotor 3/4 span

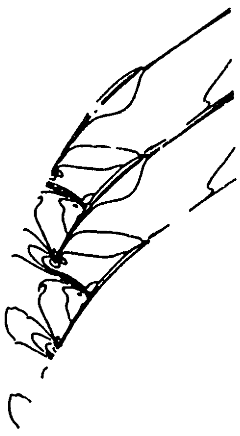

Figure C-15: Contour Mach plot - Rotor 3/4 span

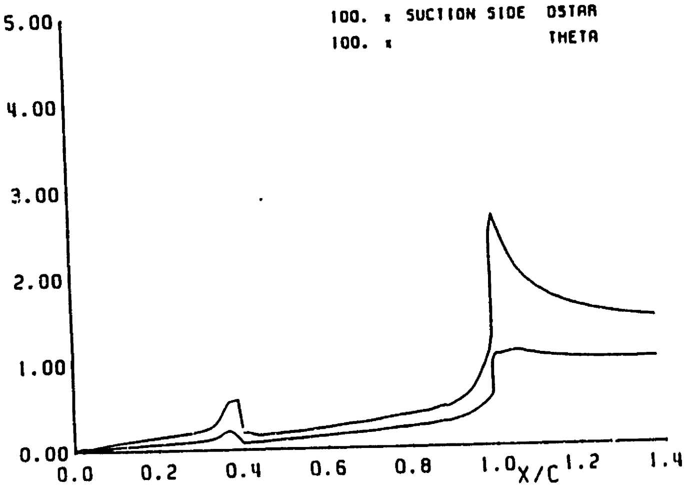

Figure C-16: Suction side boundary layer thickness - Rotor 3/4 span

Figures C-17 through C-20 show the solution found for the rotor tip.

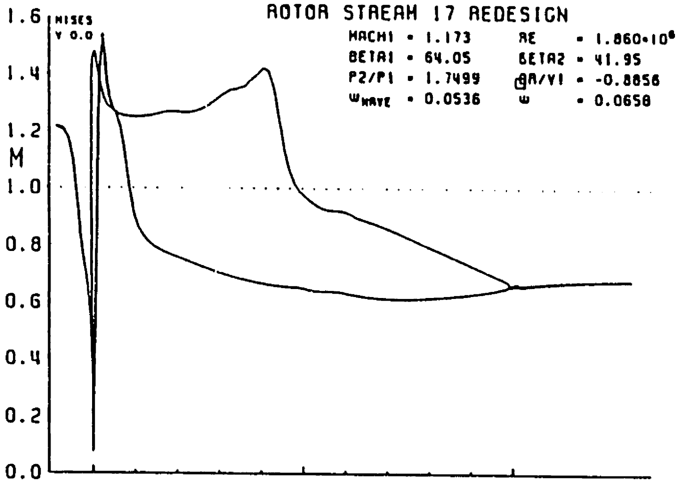

Figure C-17: Surface Mach distribution - Rotor tip

Figure C-18: Computation grid - Rotor tip

Figure C-19: Contour Mach plot - Rotor tip

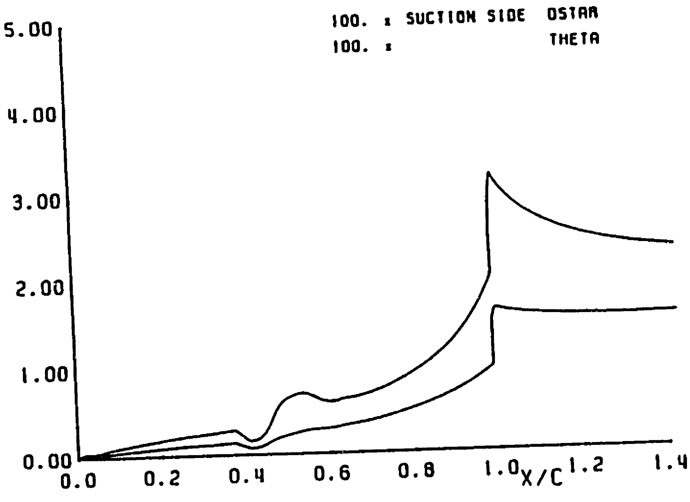

Figure C-20: Suction side boundary layer thickness - Rotor tip

Figures C-21 through C-24 show the solution found for the stator hub.

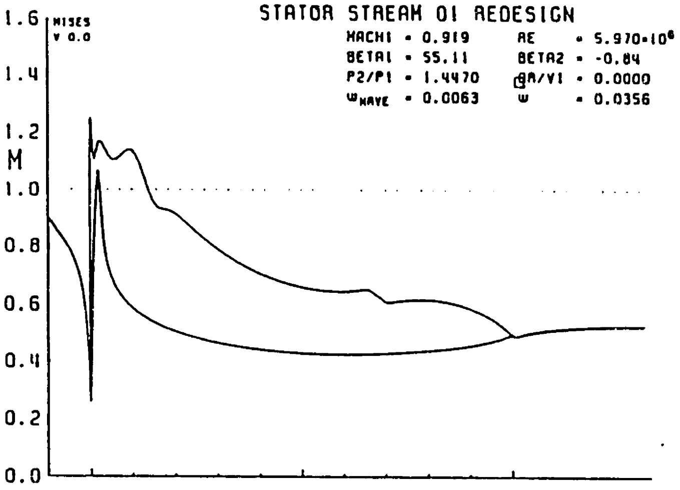

Figure C-21: Surface Mach distribution - Stator hub

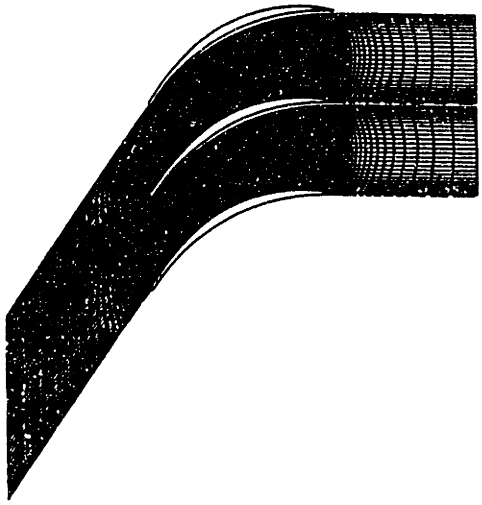

Figure C-22: Computation grid - Stator hub

Figure C-23: Contour Mach plot - Stator hub

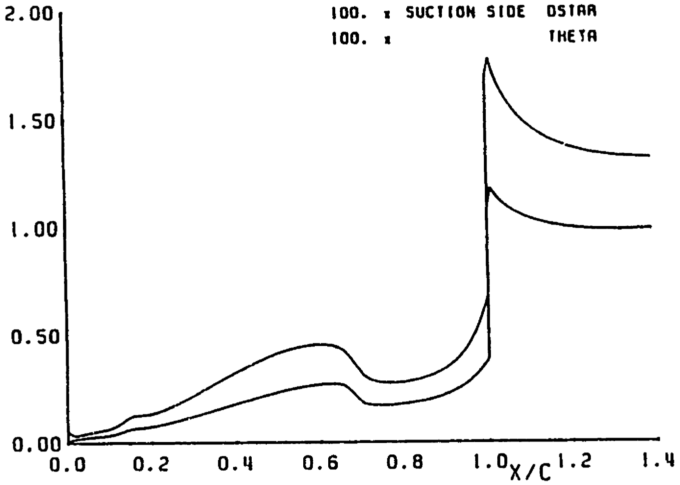

Figure C-24: Suction side boundary layer thickness - Stator hub

Figures C-25 through C-28 show the solution found for the stator 1/4 span.

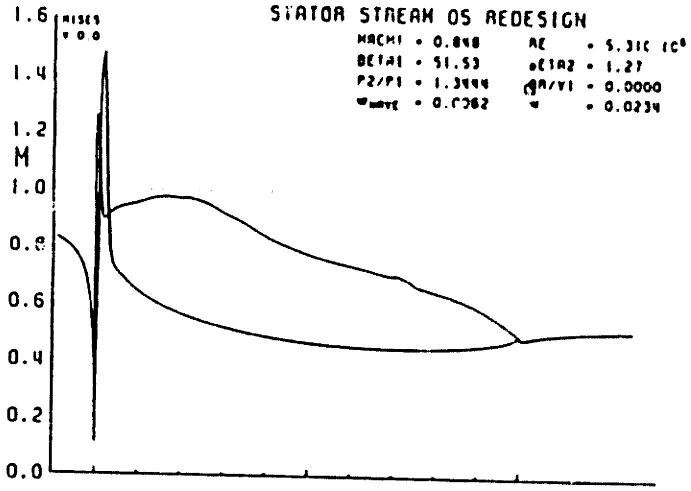

Figure C-25: Surface Mach distribution - Stator 1/4 span

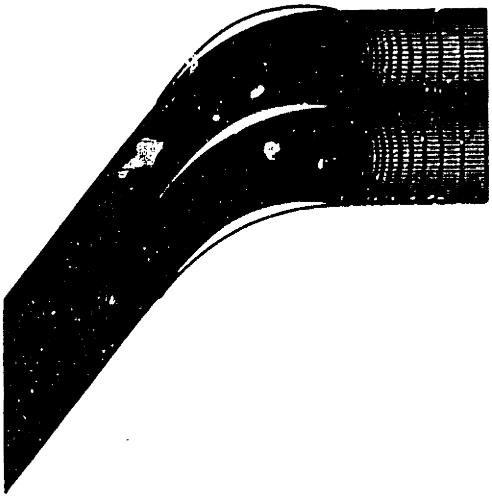

Figure C-26: Computation grid - Stator 1/4 span

Figure C-27: Contour Mach plot - Stator 1/4 span

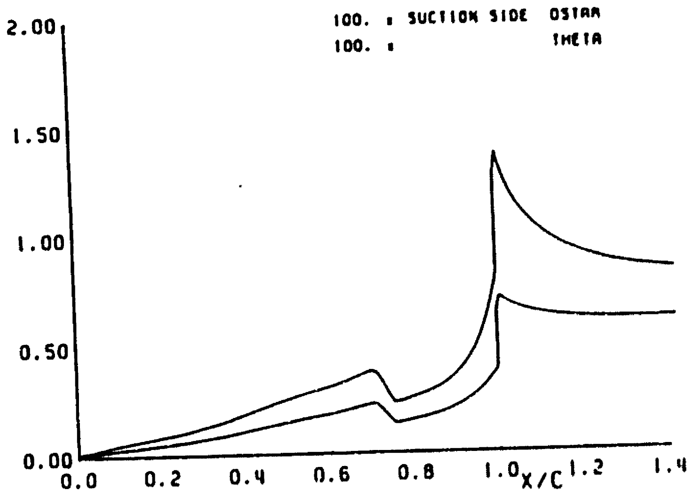

Figure C-28: Suction side boundary layer thickness - Stator 1/4 span

Figures C-29 through C-32 show the solution found for the stator 1/2 span.

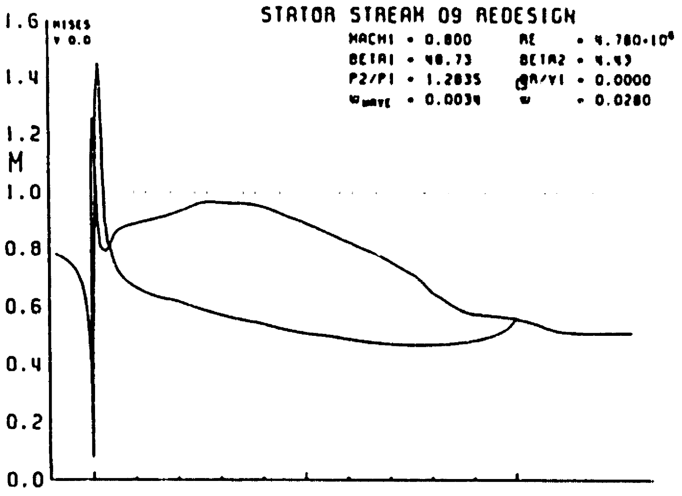

Figure C-29: Surface Mach distribution - Stator 1/2 span

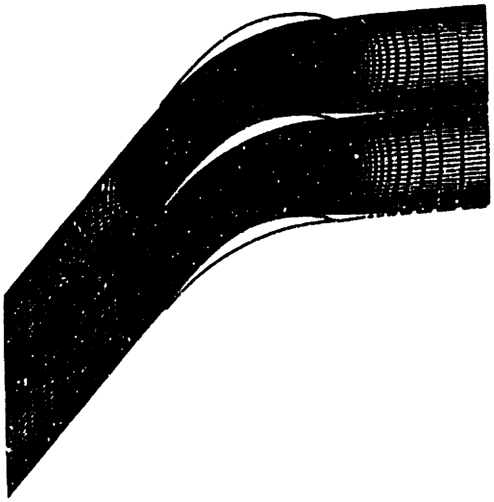

Figure C-30: Computation grid - Stator 1/2 span

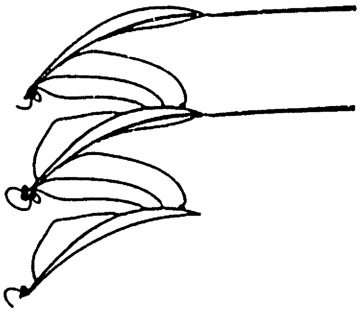

Figure C-31: Contour Mach plot - Stator 1/2 span

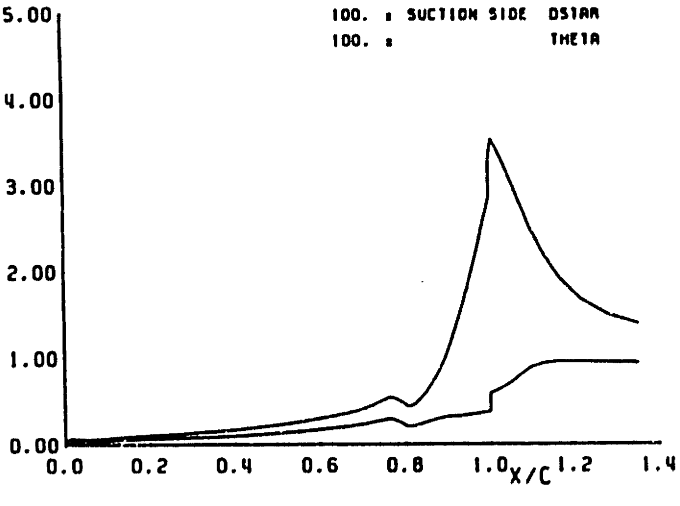

Figure C-32: Suction side boundary layer thickness - Stator 1/2 span

Figures C-33 through C-36 show the solution found for the stator 3/4 span.

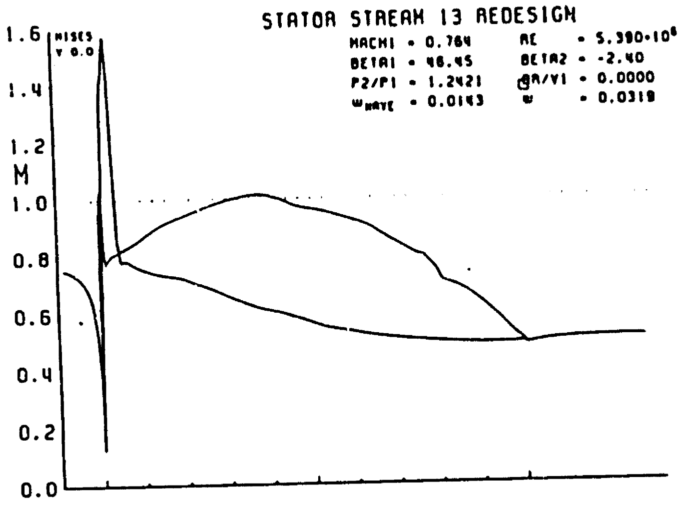

Figure C-33: Surface Mach distribution - Stator 3/4 span

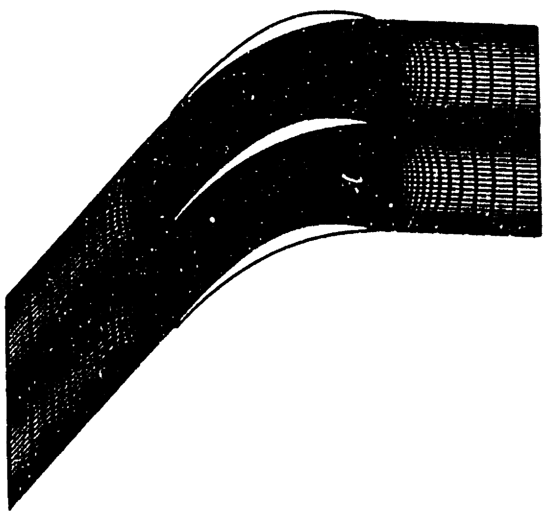

Figure C-34: Computation grid - Stator 3/4 span

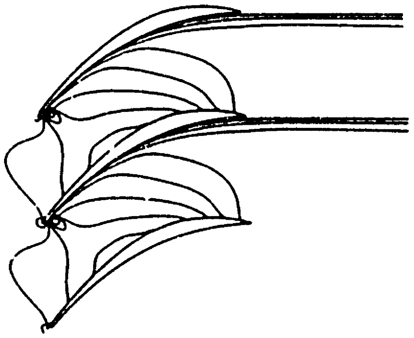

Figure C-35: Contour Mach plot - Stator 3/4 span

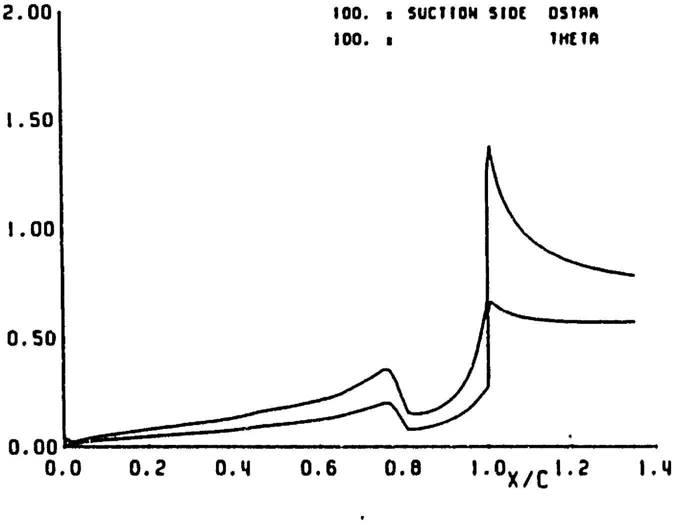

Figure C-36: Suction side boundary layer thickness - Stator 3/4 span

Figures C-37 through C-40 show the solution found for the stator tip.

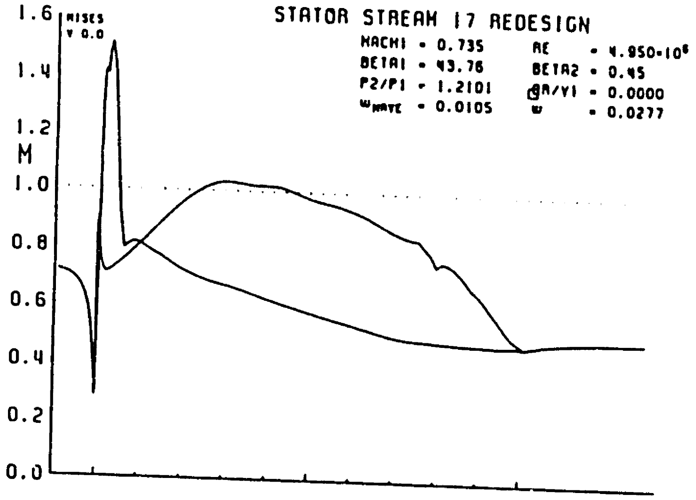

Figure C-37: Surface Mach distribution - Stator tip

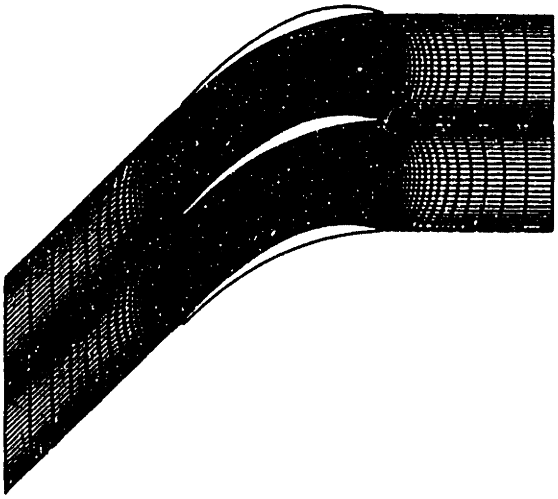

Figure C-38: Computation grid - Stator tip

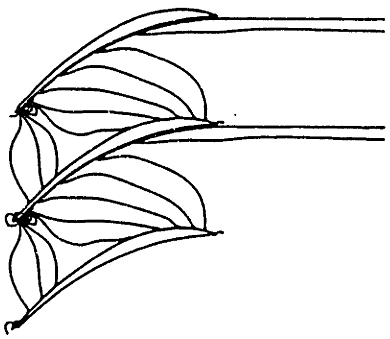

Figure C-39: Contour Mach plot - Stator tip

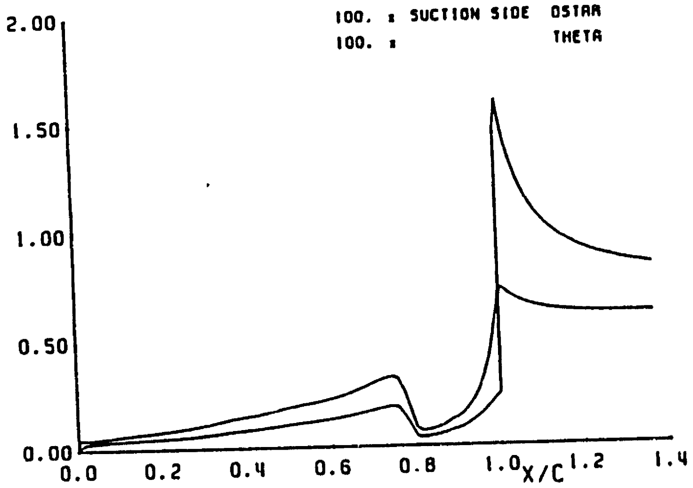

Figure C-40: Suction side boundary layer thickness - Stator tip

Bibliography

[1] W. Roland Davis and D. A. J. Millar. Through flow calculations based on matrix inversion: Loss prediction. In *Through-flow Calculations in Axial Turbomachinery*, number AGARD-CP-195 in AGARD Conference Proceedings, chapter 3. France, October 1976.

[2] Jed Dennis. *A Study of Tip Suction in Compressors.* Master's thesis, MIT, Department of Aeronautics and Astronautics, September 1993.

[3] Mark Drela. *Two-Dimensional Transonic Aerodynamic Design and Analysis Using the Euler Equations.* PhD dissertation, MIT, Department of Aeronautics and Astronautics, December 1985.

[4] Mark Drela. Personal communication, August 1994.

[5] R. M. Hearsey. A revised computer program for axial compressor design. Aerospace Research Laboratories Report ARL-TR-75-0001, Wright Patterson AFB, Dayton, Ohio, January 1975.

[6] Charles Hirsch. Computational methods for turbomachinery flows. Technical Report NPS 67-84-022, Naval Postgraduate School, Monterey, California, December 1984.

[7] Jack Kerrebrock. *Aircraft Engines and Gas Turbines.* The MIT Press, Cambridge, Massachusetts, second edition, 1992.

[8] R. J. Lougherty, R. A. Horn, Jr., and P. C. Tramm. Single-stage experimental evaluation of boundary layer blowing and bleed techniques for high lift stator

blades. Contractor Report CR-54573, Detroit Diesel Allison, Indianapolis, Indiana, March 1971.

[9] R. A. Novak. Flowfield and performance map computation for axial flow compressors and turbines. In *Modern Prediction Methods for Turbomachine Performance*, number AGARD-LS-83-1976 in AGARD Lecture Series, chapter 5. France, June 1976.

[10] A. J. Wennerstrom. Experimental study of a high throughflow transonic axial compressor stage. *ASME Journal of Engineering for Gas Turbines and Power*, 106:552–560, 1984.

[11] H. H. Youngren. Analysis and design of transonic cascades with splitter vanes. GTL Report 203, MIT Gas Turbine Lab, Cambridge, Massachusetts, March 1991.