Computer-Aided Design of Axial-Flow Fans

by

Laura M. Powers

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

APPROVED:

Dr. J. B. Jones, Chairman

Dr. H. L. Moses

Dr. A. Myklebust

August 23, 1986.

Blacksburg, Virginia

Computer-Aided Design of Axial-Flow Fans

by

Laura M. Powers

Dr. J. B. Jones, Chairman

Mechanical Engineering

(ABSTRACT)

This thesis examines the application of computer-aided design techniques to the field of turbomachinery. Specifically, the process of designing low- to medium-speed, axial-flow fans and blowers is discussed, and a Fortran program called FAN3D is introduced. The first purpose of FAN3D is to perform the aerodynamic and mechanical calculations needed to establish the basic geometry of an axial-flow fan blade. Next, geometric modeling techniques are used to model the curves and surfaces of the blade, thereby completing the geometric description of the blade. Finally, FAN3D uses the CADCD component of the CADAM (CADAM, Inc.) Geometry Interface to automatically enter the three-dimensional blade model in the CADAM database.

And to thy speed add wings

-Milton, *Paradise Lost*

Acknowledgements

I would like to express my gratitude to the following people for their help and encouragement. Each one has played a significant role in my graduate studies.

Dr. J. B. Jones, who has served as the chairman of my advisory committee. I thank him for his patience, guidance, and support. I have never left his office without gaining at least one new piece of knowledge.

Dr. H. L. Moses and Dr. A. Myklebust for serving on my advisory and examining committee. Their guidance and support has been invaluable.

My fellow graduate students for their technical guidance, moral support, and comradery. In particular, Tom Popernack, Doug Roach, Greg Sherman, Harinder Singh Oberoi, Steve Wampler, and Bernie Maloney are constant sources of support and laughter.

Mrs. Jan Riess, whose smile and understanding word make Randolph Hall seem brighter.

Finally, I am indebted to my family for their unfailing love and support. I value their faith in my ability to achieve anything I can imagine.

Table of Contents

List of Illustrations

List of Tables

Nomenclature

A	area
c	chord (see Figure 9)
C	circulation
d_s	specific diameter
D	diameter
e	compressibility
g	blade clearance gap
h	enthalpy
H	blade span, head, blade shape factor(see Figure 5)
i	incidence angle (see Figure 5)
m	mass flow rate
n_s	specific speed
N	rotational speed
p	pressure
P	power
Q	volumetric flow rate
r	radius
R	degree of reaction, gas constant
s	spacing (see Figure 5)
T	temperature, torque
U	blade speed (see Figure 6)
V	absolute velocity (see Figure 6)
w	work per unit mass
W	relative velocity (see Figure 6)

α	absolute flow angle (see Figure 5)
β	relative flow angle (see Figure 5)
β'	blade angle (see Figure 5)
γ	stagger angle (see Figure 5)
δ	deviation angle (see Figure 5)
δu_x	blade loading factor
η	efficiency
ζ	loss coefficients
κ	specific heat ratio
λ	work done factor
λ_∞	mean vector angle
μ	dynamic viscosity
π	similarity parameter
φ	head coefficient
Φ	flow factor
ρ	density
σ	solidity (s/c)
θ	camber angle
θ_2/c	wake momentum thickness
ω	angular velocity
$\bar{\omega}$	total loss coefficient

Subscripts

a	annulus
c	critical
ew	end wall
i	inside (hub)
ie	ideal Euler
m	mean
max	maximum
min	minimum
o	outside (tip)
p	profile
r	relative, ratio (as in pressure ratio)
s	isentropic, secondary, specific
t	total
tc	tip clearance
x	axial direction
θ	tangential direction
1	rotor inlet
2	rotor exit

NOTE:

The nomenclature on mechanical strength is listed in that chapter.

Variables in boldface like (**P**) denote a vector.

Introduction

Computers are conventionally used as an analytical tool in the field of turbomachinery. When flow equations cannot be solved analytically, computers offer numerical solution techniques. Finite difference and finite volume computational methods model complex flow fields. Experimental data are reduced and mathematical models are created using computers.

However, in addition to these analytical applications, computers can significantly aid the designer of turbomachinery. As in any other engineering design, the design of turbomachinery is a complex iterative task. Initially, the simplest design is created. More refined designs follow as assumptions are modified and compromises are struck between concerns like structural integrity and thermodynamic efficiency. In this process, computers can both reduce a designer's work and extend his capabilities from the first design of a turbomachine through its manufacture. The purpose of this thesis is to demonstrate how computer-aided design techniques can be applied to a "classical" design method for turbomachinery.

The focal point of this thesis is the design of axial-flow fans and blowers. The design method, as documented in the Fortran 77 computer program FAN3D and supported by the CADAM geometry interface, is intended to be a versatile design tool, rather than a finished program to design and optimize any axial-flow fan. A fan designer should be able to adapt, modify, or extend this method to meet his current geometrical, aerodynamic, and graphical requirements or limitations. The primary aim of this thesis is to examine the design of low- to medium-speed fans, but the design development can also be regarded as the initial step in a design method for high-speed, high-performance axial-flow fans.

With these goals in mind, the intended audiences for this discussion are those people who are interested in:

- fluid mechanics and its applications
- computer-aided design and its applications
- axial-flow fan design and its enhancement via computers

This thesis follows a chronological development of the design of axial- flow fans. That is, the sequence of discussion follows the same path as would a designer, from the initial decision to use an axial-flow piece of equipment to what may be called the "detailed design." Both aerodynamic and computer concepts are introduced as needed in the development.

Classifications of Turbomachines

According to Dixon [8], the word *turbo* originated from the Latin word *turbinis* which means to spin or whirl. Thus, a turbomachine is a piece of spinning or whirling equipment. More specifically, turbomachines are defined as those devices in which energy is transferred either to or from a continuously moving fluid by the dynamic action of a moving blade row. Essentially, the rotating blade row, or rotor, changes the stagnation enthalpy of the fluid moving through the row by work done either on or by the fluid.

Turbomachines may be classified according their production or consumption of work. When positive work is done by the fluid, the turbomachine produces power by expanding the fluid to a lower pressure or head. Turbomachines of this type are usually called turbines. When negative work is done by the fluid; that is, when work is done on the fluid, the turbomachine consumes power in order to increase the fluid pressure or head. Fans, blowers, and compressors all belong in this category.

In addition to the classification of turbomachines according to their production or consumption of power, turbomachines may be independently classified by the general path followed by the fluid as it passes through the machine. In a radial-flow turbomachine, the path of the fluid flow is primarily perpendicular to the axis of rotation of the turbomachine. Similarly, the fluid flow in an axial-flow turbomachine is primarily parallel to the axis of rotation. Finally, in a mixed-flow machine, both axial and radial velocity components are present in significant amounts at the rotor exit.

Using the classifications developed thus far, this thesis focuses on axial-flow turbomachines which consume power in order to increase fluid pressure. But how do fans, blowers, and compressors differ? Using the guidelines described by Pollack [25], a fan is a machine that produces pressure rises

up to about fourteen kilopascals. When the pressure rise is between 14 and 70 kPa, the machine is called a blower, and above 70 kPa, it is a compressor.

Finally, the subject of this thesis, axial-flow fans, has been further classified by the National Association of Fan Manufacturers as tube-axial and vane-axial. Tube-axial-flow fans basically consist of a propellor enclosed in a cylinder which collects and distributes air. The path of the air leaving the rotor is helical or screwlike, as shown in Figure 1. Vane-axial-flow fans are essentially tube-axial-flow fans with guide vanes at the rotor exit to eliminate the helical flow path, as shown in Figure 2. The addition of guide vanes usually improves the efficiency of the fan.

The following development of rotor design is applicable to the design of rotors in either tube-axial- or vane-axial-flow fans.

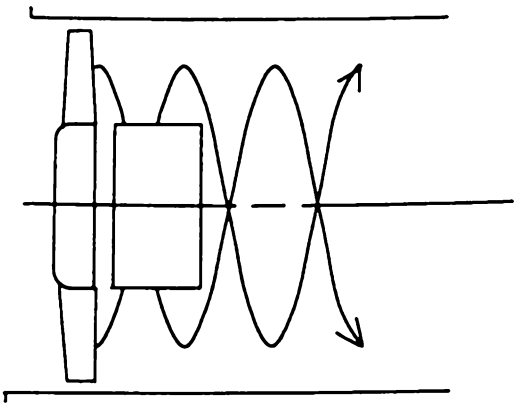

Figure 1. Schematic of Tube-Axial-Flow Fan

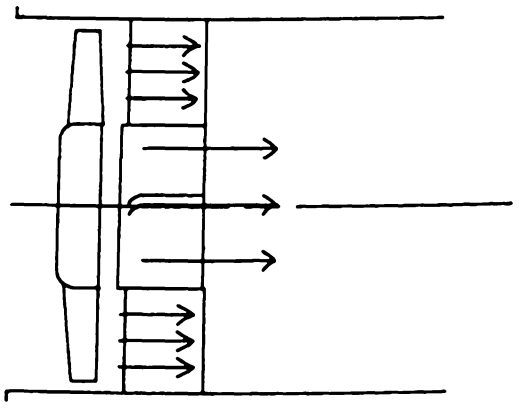

Figure 2. Schematic of Vane-Axial-Flow Fan

Literature Review

Since the purpose of this thesis is to investigate the enhancement of axial-flow turbomachine design using computer-aided design methods, two areas of literature review are necessary. First, the development and current status of methods for designing axial-flow turbomachines will be examined, followed by a discussion of the development of computer-aided design and its applications to turbomachinery. Neither review should be regarded as an exhaustive evaluation, but rather as a foundation for the discussions that follow.

Axial-Flow Turbomachinery

Both Dixon [8] and Horlock [12] attribute the original idea of reversing a turbine in order to increase fluid pressure or head to Sir Charles Parsons. In 1901, Parsons was granted United Kingdom Patent No. 3060 entitled "Improvements in Compressors and Pumps of the Turbine Type." Parsons built some of these machines for use in blast furnace work with pressure rises between 10 and 100 kPa. Initially, the maximum efficiency of a reversed turbine was only forty percent. Even with later blading section improvements, Parsons was still limited by maximum efficiencies of less than fifty-five percent. As a result, axial-flow machines were abandoned in favor of multistage centrifugal machines whose maximum efficiencies ranged between seventy and eighty percent.

Further development of axial-flow fans and compressors was delayed until 1927 when A. A. Griffith outlined the basic principles of airfoil theory for axial compressor and turbine design for the Royal Aircraft Establishment. The subsequent history of the axial-flow fan or compressor is linked to the development of the aircraft gas turbine, as recorded in 1945 by Horlock [12].

The early difficulties in achieving acceptable efficiencies in compressors and fans stem from the difference in the way a fluid flows through a compressor or fan as opposed to a turbine. A turbine produces power by expanding a fluid to a lower pressure or head. In this expansion process, the fluid accelerates relative to each blade. In contrast, a compressor or fan consumes power in order to increase the fluid pressure or head. In this case the fluid flow is decelerated.

The key to the greater compressor losses is the adverse pressure gradient responsible for retarding or decelerating the flow. In the presence of such a gradient, the boundary layers on the surfaces of the blades grow faster. The associated shear stresses and drag forces on the blade surface also grow, thereby increasing the total pressure losses. Eventually the slowing of flow in the boundary layers may lead to separation of the boundary layers from the blade surfaces, creating regions of reversed flow. The airfoil is then said to be stalled.

The solution to compressor or fan performance problems associated with a large adverse pressure gradient is to restrict the rate of deceleration or turning in the blade passages. While this limits the maximum pressure rise available through a compressor or fan rotor, greater pressure rises are possible by building multistage machines. The collective flow losses through a series of low-pressure-rise units can be much less than the losses through a single unit producing the entire pressure rise. As a result, maximum efficiencies today approach ninety percent, while many engineers believe that the full potential for increasing efficiency and pressure ratio in axial-flow compressors and fans has yet to be achieved.

Almost a century after Parsons built his first axial-flow compressor, the literature discussing how to design, optimize, and manufacture axial-flow turbomachines has grown to enormous proportions. Much of the development contained in this thesis relies heavily on the design methods described by O. E. Balje [2] and R. Allan Wallis [32]. As with many design areas, several "schools of thought" flourish in turbomachine design. Balje is a leader in the German school of turbomachine design, while Wallis is British. In some cases this difference causes slight variations in their development of turbomachine design. For example, Balje introduces similitude quickly in

his development, and relies on dimensionless parameters for many calculations like estimation of losses. Wallis barely mentions this concept, instead performing most calculations in a dimensional form. In each step of the following development, exact references are cited, and any differences of opinion are noted.

Computer-Aided Design

Compared to established fields like turbomachinery and its parents--thermodynamics and fluid mechanics-- computer-aided design is a young field. Even the simplest computer-aided system must perform huge numbers of computations in order to perform graphical operations like hidden-line removal or shading. Thus, it is highly dependent on advances made in computer hardware technology to improve its computing speed and memory capabilities.

Krouse [16] and Nystrom [22] provide historical perspectives of both the development of computers and computer-aided design. In the 1940's, digital computers were large electromechanical devices using clicking relays to perform computations. Improvements in the late 1940's and early 1950's included the replacement of relays with vacuum tubes and the advent of data storage devices like disks and drums and magnetic cores for program memory. Later in the 1950's, the transistor replaced the vacuum tube, resulting in a much more compact, reliable, faster computer. And finally, the introduction of integrated circuits in the early 1970's helped to finish setting the stage for the computer-aided revolution. Not only has integrated circuit technology reduced computer size while increasing speed, it has more significantly reduced the cost of computers. In this decade, computer-aided design has become cost-effective for the small and medium engineering firms as well as the traditional giants.

As was previously mentioned, the continuing development of computer-aided design is highly dependent on corresponding advances in computer technology. One of the primary causes of this

dependence is the need for easy communication between the designer and the computer. Previously, a person entered data into a computer using punched cards or paper tape. Experience in computer programming and operation was prerequisite. Now, the communication link has been simplified so that much communication takes place using on-screen pictures. While the need for experience in computer programming has not been completely eliminated, often a high-level language like Fortran is now used in place of assembly language, and training in CAD software has replaced the training in computer operation.

One of the earliest developments in interactive computer graphics was the Sketchpad Project led by Ivan Sutherland at Massachusetts Institute of Technology in the early 1960's. Using a cathode ray tube (CRT) monitor, data was entered using a lightpen at the scope. By entering point coordinates and following simple computer requests, a designer could generate lines, circles, arcs, and other pieces of an engineering drawing. Once entered, the data could be used to create modified drawings, to produce hardcopies, or as input to an analytical program.

Several other interactive CAD systems were also developed in the 1960's, primarily by large companies for in-house design work. Surprisingly, some of the first computer-aided design (CAD) systems were developed by the same companies who fueled work in turbomachine design, the aircraft industry. Initially, a CAD system was primarily a two-dimensional drafting tool. A designer could use the computer, instead of pencil and paper, to generate engineering drawings. Later, many systems were expanded to include computational features like finite element analysis and generation of numerical-control instructions. Eventually the speed and memory capabilities of CAD were improved to the extent that three-dimensional CAD systems were introduced. At this point, computer-aided design became a more appealing design tool for turbomachinery where three dimensions are essential for even the most simple design concept.

Ozell [24] provides an excellent background on computer-aided design of turbomachinery. The primary difference between Ozell's design methods and those described here is the way in which the computer draws the turbomachine geometry. In Ozell's case, the designer must enter the three-

dimensional coordinates of points lying on the blade surface. A smooth model of a blade would require many points to be manually entered. In contrast, this thesis introduces the use of a geometry interface between the computation of geometry and its graphical representation. The chore of keying in point coordinates is eliminated.

Design Aspects

The design of a turbomachine is a complex iterative procedure. Thermodynamic, aerodynamic, technological, structural, and economic aspects all demand the designer's attention, even in the initial design stages. The process is further complicated by the strong interactions among some of these aspects. Simplifications are necessary. Macchi [18] recommends that the procedure be simplified by assuming that the thermodynamic and fluid-dynamic aspects of the design are independent. He further recommends that all other aspects be translated into constraints used to modify or limit the design. For example, manufacturing abilities may limit the thickness of a blade's trailing edge, or mechanical stresses may limit peripheral speed to reduce centrifugal stresses.

In order to meet these design criteria, the physical process occurring as the fluid flows through the turbomachine must be accurately described in a mathematical model. The equations representing this flow are fundamentally three-dimensional, nonlinear, and unsteady. The solution to the set of full laminar equations with full boundary conditions represents a complex problem--from both computational and mathematical viewpoints. In the case of turbulent flow models, the equations themselves are incomplete and no analytical solution is yet possible. Simplifications and assumptions are again necessary. Typically, the initial design assumes one-dimensional flow, and the turbomachine is designed at a single radial position. More refined designs follow as assumptions are modified.

In addition to the complexities of the flow field, the components of the turbomachine itself present design challenges. Some components rotate; some are stationary. The blading geometry is three-dimensional because the cross-sectional geometry as well as the camber angles change from blade hub to blade tip. Boundary layers grow on the blades and annulus walls. Wakes form at the trailing edges of the blades. Flow leaks around the blade tips. Since turbomachine components

must be designed from a three-dimensional viewpoint, losses due to this geometry--boundary layer, wake, and leakage losses-- ideally must also be evaluated including the effects of three dimensions.

Yet another complication in the design of turbomachinery is the reliance on empirical relations to bridge gaps between theory and computation. Often these empirical relations must be assumed applicable to situations other than those for which they were derived. For example, loss relations for flow through a flat-plate, stationary cascade of blades might be applied to a cambered airfoil in a rotating turbomachine. Accurate prediction of the performance of a turbomachine depends on judicious application of empirical relations.

Finally, Shepherd [27] classifies axial-flow turbomachines as the most difficult type to design. In cases where a designer has a choice between types of flow path required in the turbomachine, axial-flow turbomachines often offer better performance than corresponding radial- or mixed-flow machines. However, small deviations from the optimum can be disastrous to performance. An axial-flow machine can often be conservatively designed for operation over a wider range of conditions, but this may negate any advantages over a radial- or mixed-flow machine.

When is an axial-flow machine preferable to a radial- or mixed-flow machine? In what cases is more than one machine type applicable? The following section introduces the concepts of specific speed and specific diameter, and uses these to help a designer decide when an axial-flow turbomachine is applicable.

Dimensional Analysis and Similitude

Definitions

Two useful tools for turbomachinery design are dimensional analysis and similarity (similitude). *Dimensional analysis* is the formal procedure of reducing a group of independent variables that describe a given physical situation to a smaller group of dimensionless parameters. Related to dimensional analysis is the concept of *similarity* or *similitude* in which characteristics of one turbomachine are compared with another. The three types of similarity are:

- *Geometrical Similarity* where the ratios of dimensions of corresponding parts are equal throughout the two machines.

- *Kinematic Similarity* where the velocity triangles at corresponding points in the two flow fields are similar. Kinematic similarity implies geometric similarity.

- *Dynamic Similarity* where the ratios of the forces acting on the fluid elements are equal. Dynamic similarity implies kinematic (and thus geometric) similarity.

Thus, one of the foremost advantages of dimensional analysis and similarity is the ability to predict the performance of one turbomachine from the known performance of a similar machine. Experimental tests on machine A at one speed with one fluid can help predict the characteristics of a similar machine B, operating at a different speed with a different fluid. In addition to the comparison of performance between two fullsize turbomachines, similarity enables the prediction of a prototype's performance based on tests of scale models of the machine. Such tests save both time and money.

Dimensional analysis also permits economical graphical presentation of experimental results. Families of curves expressed in terms of independent variables can often be described in a single curve in terms of dimensionless groups of independent variables (similarity parameters). Finally, dimensional analysis helps in the determination of the most suitable type of machine on the basis of maximum efficiency, head, rotational speed, and mass flow.

Determination of Similarity Parameters

A number of different methods of dimensional analysis can yield the dimensionless groups of independent variables or similarity parameters. These methods include:

- Examination of the basic flow equations. Some quantities like the Reynolds number can be derived from the fluid flow equations.
- Exact solution of simplified cases of the flow equations.
- Buckingham Pi-Theorem.

The Buckingham Pi-Theorem is a formal procedure for deriving dimensionless groups by arranging the variables with unknown powers in groups. That is, a physical quantity Q_1 is interrelated with other quantities Q_2, Q_3, ... Q_n by

$$Q_1 = f(Q_2, Q_3, \dots, Q_n) \tag{1}$$

where $Q_1, Q_2, Q_3, \dots Q_n$ all represent independent variables describing a physical situation or process--like fluid flow through a turbomachine.

Introducing the π-form, equation 1 may be written as

$$\pi_1 = \varphi(\pi_2, \pi_3, \cdots, \pi_{n-d}) \tag{2}$$

where d is the number of primary dimensions (see below) and each π-term is a product of the form

$$\pi = Q_1^a Q_2^b \cdots Q_n^x \tag{3}$$

The powers a,b,...,x are then determined by the constraint that each π-term must be dimensionless. Thus, some powers may be zero. Balje [2] and Shepherd [27] both describe application of the Buckingham Pi-Theorem in detail.

How many independent similarity parameters or π-terms are required to describe a physical process? The required number of similarity parameters equals the number of independent variables in the physical process minus the number of primary dimensions. Typically nine independent variables are defined for turbomachines. These variables are listed in Table 1. With three primary dimensions (mass, length, and time), six independent similarity variables are required to describe flow through a turbomachine result.

It is important to emphasize that only six independent similarity parameters can be found for turbomachines handling a single-phase, compressible medium. No new parameters can be discovered by manipulating the existing set of nine independent variables. However, the choice of six parameters is somewhat arbitrary. Table 2 defines the six used by Balje [2]: efficiency, Reynolds number, Laval number, specific heat ratio, specific speed, and specific diameter. Other sources like Shepherd [27] replace the last two parameters with the head coefficient and flow coefficient, as also shown in Table 2. The remainder of this development will adhere to Balje's set of parameters.

Application of Specific Speed and Specific Diameter

As previously mentioned, dimensionless parameters are useful in design and machine selection. In particular, specific speed and specific diameter are often used to aid the selection of flow-path (axial, mixed, or radial). Specific speed is a measure of flow-handling capacity and pressure-rise capacity as a function of rotational speed, but not rotor diameter. Likewise, specific diameter describes flow- and pressure-rise capacities in terms of the rotor diameter instead of the rotational speed.

Csanady [7] provides recommendations for selecting the flow-path based on specific speed, as shown in Figure 3. In addition, Csanady illustrates how specific speed can be used to select the specific diameter using the Cordier diagram in Figure 4. While this diagram represents a minimization of the scatter of data points, it can be expected that a turbomachine designed using this information would have an acceptable if not optimal rotor diameter.

For incompressible, single-phase media, the set of similarity parameters can be further reduced. The Laval number and the ratio of specific heats become irrelevant. They are replaced by a single term expressing cavitation sensitivity, like suction specific speed.

Table 1. Basic Turbomachine Variables

Symbol	Dimensions	Parameter
ω	T^{-1}	rotational speed
D	L	diameter
Q	L^3T^{-1}	volumetric flow
ρ	ML^{-3}	density
H	L	head
μ	$ML^{-1}T^{-1}$	dynamic viscosity
P	ML^2T^{-3}	power
e	$M^{-1}LT^2$	compressibility
κ	*none*	ratio of specific heats

Note:
M = mass
L = lenth
T = time

Table 2. Turbomachine Similarity Parameters

	Symbol	Parameter	Definition
Balje	η	efficiency	$\frac{gH}{w_{in}}$
	Re	Reynolds number	$\frac{\rho VD}{\mu}$
	La	Laval number	$\frac{V}{V_c}$
	κ	specific heat ratio	$\frac{C_p}{C_v}$
	n_s	specific speed	$\frac{\omega\sqrt{Q}}{(gH)^{3/4}}$
	d_s	specific diameter	$\frac{D(gH)^{1/4}}{\sqrt{Q}}$
Shepherd	η	efficiency	$\frac{gH}{w_{in}}$
	Re	Reynolds number	$\frac{\rho VD}{\mu}$
	La	Laval number	$\frac{V}{V_c}$
	κ	specific heat ratio	$\frac{C_p}{C_v}$
	φ	flow coefficient	$\frac{Q}{\omega D^3}$
	ψ	head coefficient	$\frac{gH}{\omega^2 D^2}$

Note: $n_s = \frac{\varphi^{0.5}}{\psi^{0.75}}$ and $d_s = \frac{\psi^{0.25}}{\varphi^{0.5}}$

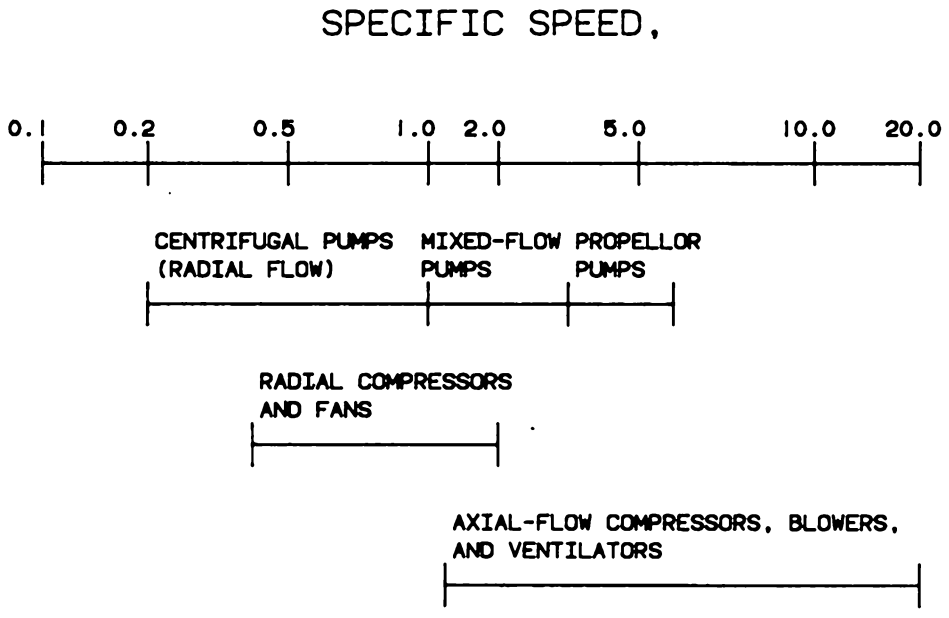

Figure 3. Recommended Flowpath Based on Specific Speed

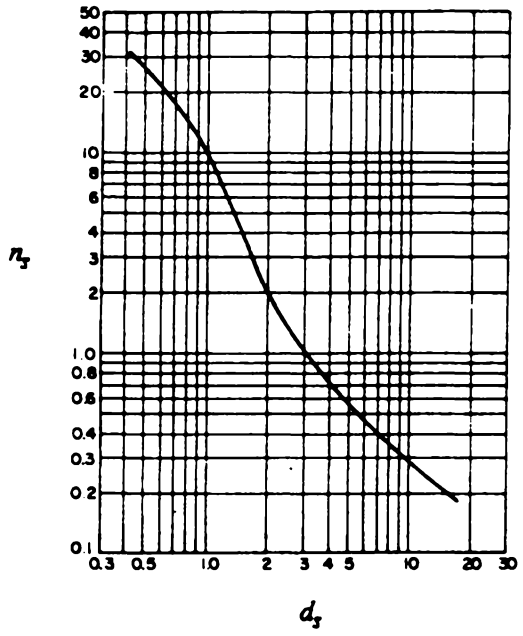

Figure 4. Recommended Specific Diameter Based on Specific Speed [7]

Meanline Design

Introduction

The basic specifications required in the design of axial-flow fans, blowers, and compressors include the mass flow rate, the pressure rise across the turbomachine, and the inlet conditions such as temperature and pressure. These quantities may be regarded as fixed. A designer additionally sets other parameters like the hub radius, hub-to-tip radius ratio, and the rotational speed. These quantities are partially governed by other constraints like mechanical strength or the intended application of the turbomachine; however, more than one hub radius, hub-to-tip ratio, or rotational speed may be permissible. Ultimately the designer chooses the optimum combination of specifications.

Often the first step in the design of a turbomachine is a radial two-dimensional analysis performed at one position on the blade. In order for this analysis to be even generally indicative of the true performance of the turbomachine, the position for analysis must be wisely chosen. Balje [2], Wallis [32], and most other sources recommend analysis at the meanline because it represents average flow conditions better than either the hub or tip of the blade. The meanline or mean square radius is defined as

$$r_m = \sqrt{\frac{(r_o^2 - r_i^2)}{2}} \tag{4}$$

where r_o is the rotor tip radius and r_i is the rotor hub radius.

What is the result of a meanline design? In simplest terms, it is a determination of the average flow deflection in a turbomachine. The inlet and exit flow angles and blade angles are calculated at the mean radius. Since the flow is assumed two-dimensional, the meanline flow deflection and losses

are consequently assumed to be representative of the whole blade, and the efficiency is calculated based on losses at the mean radius only.

Even though a meanline analysis may be simple, it is important. It helps to determine a set of potentially useful flow geometries, the number of stages, the overall turbomachine dimensions, and the general range of efficiency. Macchi [18] asserts that a properly-designed axial-flow turbomachine may have its performance predicted quite accurately by a meanline analysis--within one or two percent of more refined (and more detailed) methods. In order to achieve these accuracies, Macchi stipulates that the loss correlations must be carefully calculated so that the loss coefficients include all relevant geometric and kinematic variables. It is Macchi's opinion that the basic performance range is determined by the meanline analysis. More refined design methods may optimize the design, but will not greatly alter the efficiency.

Macchi's high opinion of meanline analysis in turbomachine design should not undermine the importance of accurate blade design or detailed flow models. The basis for his prediction of efficiencies banded by one or two percent deviations depends on accurate calculation of the loss coefficients, and if any area of turbomachine design is the current subject of concentrated research, it is the theory and computation of the loss mechanisms. But even when Macchi's claims are tempered with these difficulties in estimating losses, they are still noteworthy. The initial meanline design of a turbomachine becomes the input data for subsequent designs, and these refined models are only as good as their foundation--the meanline analysis.

Assumptions

The following assumptions are made in the meanline design:

- The radial component of velocity is zero throughout the blade passage.

- The axial component of velocity is the same at the rotor inlet and exit.
- At design conditions, the inlet flow angle equals the blade angle.
- The flow has no swirl at the rotor inlet.
- The flow is incompressible.

Two-Dimensional Cascades

Balje [2] describes two general analysis techniques typically used in designing pumps, compressors, and fans. A technique based on channel arguments is usually applied to centrifugal machines while a technique based on cascade arguments is usually applied to axial machines. This development is based on theory and tests on stationary, two-dimensional cascades of blades. A general axial-flow cascade arrangement is shown in Figure 5. The present assumption that the inlet flow angle β_1 equals the inlet blade angle β'_1 causes the incidence angle i shown in this figure to be zero.

Typical velocity triangles for the inlet and exit of an axial-flow rotor are shown in Figure 6. The flow enters the blade passage with an absolute velocity V_1 at some angle α_1. The entering velocity and angle are prescribed by upstream conditions. Relative to the moving blades, the flow velocity is W_1 at an angle β_1. The relative and absolute flow velocities are related by the blade speed U. Specifically, the absolute velocity V is the vector sum of the relative velocity W and the blade speed U . That is,

$$\mathbf{V} = \mathbf{W} + \mathbf{U} \tag{5}$$

This graphical presentation of this vector relationship is known as a velocity triangle. Following the prescribed assumption, the inlet velocity in this analysis is completely in the axial direction, or

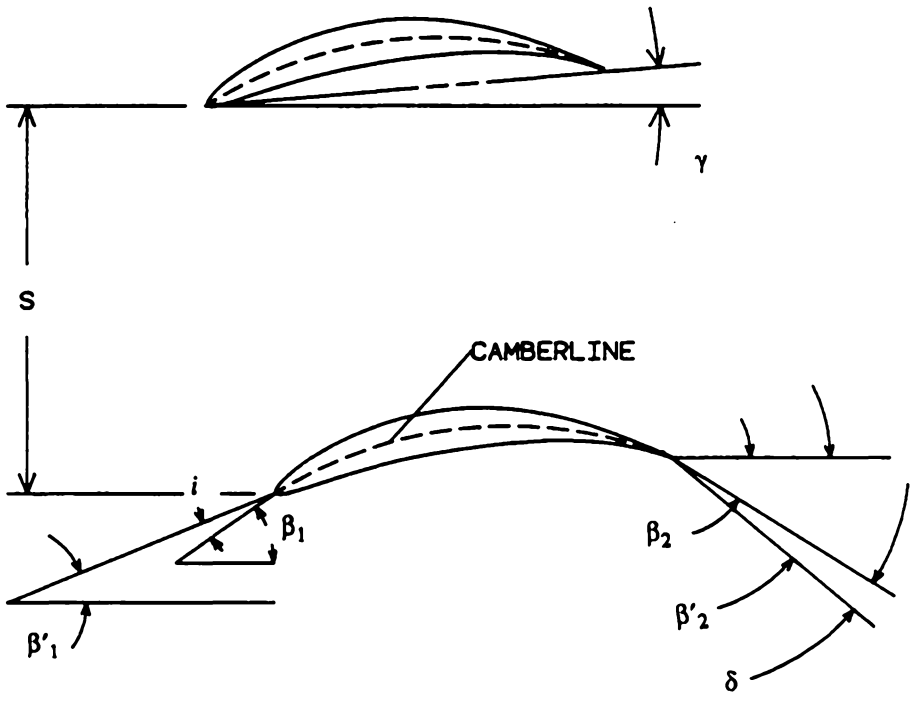

Figure 5. Cascade Nomenclature

$$V_1 = V_{x1} \tag{6}$$

so that $\alpha_1 = 0$. Figure 6 also illustrates the assumption of constant axial velocity at the rotor inlet and exit.

As the fluid flows through the blade passage, it changes direction from β_1 to β_2. The result of this change is a net force acting on the blade and resulting from the pressure difference on the two sides of the blade. The side of the blade with the higher pressure is known as the *pressure side* while the side with the lower pressure is called the *suction side*.

Flow Equations

The following equations describe the fluid flow through an axial-flow fan. The equations are presented in simplified form according to the stated assumptions.

Total Pressure:

The total pressure or isentropic stagnation pressure is defined as the pressure a fluid would have if it were stopped reversibly with no heat transfer. For incompressible flow, the total pressure p_t may be expressed as

$$p_t = p + \frac{\rho V^2}{2} \tag{7}$$

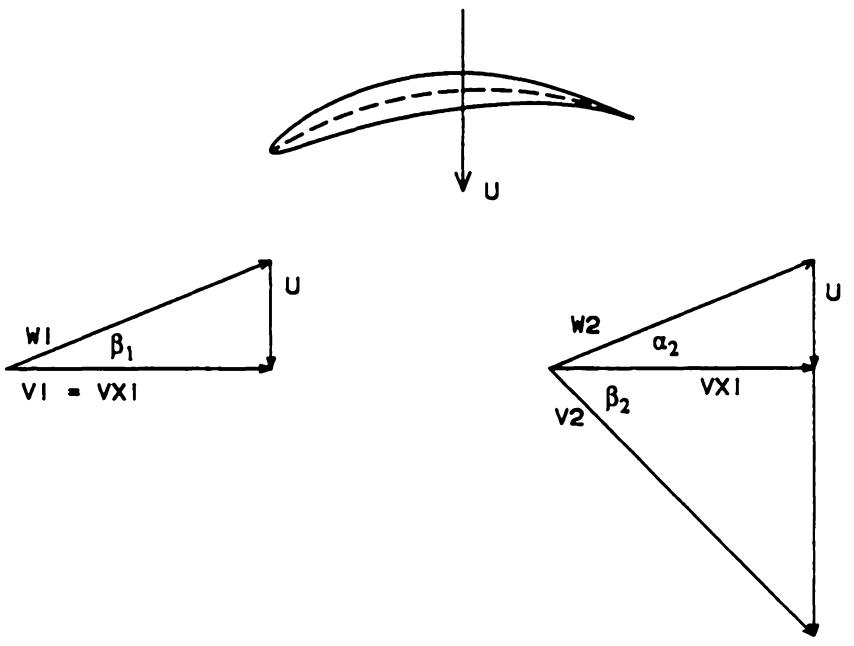

Figure 6. Axial-Flow Fan Velocity Triangles

Total head

Head is defined as pressure or presssure difference divided by the product of density and gravitational acceleration. It has the units of length and represents the height of a column of the same fluid which would exert the same pressure. Head and total head are defined in terms of pressure and total pressure respectively. The change in total head as a fluid passes from point 1 in a turbomachine to point 2 is

$$\Delta H_t = \frac{P_2 - P_1}{\rho g} + \frac{V_2^2 - V_1^2}{2g} \tag{8}$$

or

$$\Delta H_t = \frac{\Delta P_t}{\rho g} \tag{9}$$

Conservation of Mass

Assuming that the flow is uniform (except for boundary layers) and two-dimensional, the mass flow rate $\dot{m}$ through an axial-flow turbomachine may be expressed as

$$\dot{m} = B\rho_1 V_{x1}\pi(r_o^2 - r_i^2) \tag{10}$$

where B is a blockage factor accounting for the reduction in available flow area due to boundary layers on the hub and casing. A typical value for the blockage factor is 0.95. V_x is the axial component of the absolute velocity.

The First Law of Thermodynamics

Assuming a simple control volume and one-dimensional flow, the work done by the fluid (per unit mass) is

$$w = q + h_1 - h_2 + \frac{V_1^2}{2} - \frac{V_2^2}{2} + gz_1 - gz_2 \tag{11}$$

Normally the flow in a turbomachine is assumed adiabatic with negligible changes in potential energy. The equation reduces to

$$w = h_1 - h_2 + \frac{V_1^2}{2} - \frac{V_2^2}{2} \tag{12}$$

When the total enthalpy is defined as

$$h_t = h + \frac{V^2}{2} \tag{13}$$

the equation becomes

$$w = h_{t1} - h_{t2} \tag{14}$$

and the power P is

$$P = \dot{m}w \tag{15}$$

Moment of Momentum

If a fluid flows steadily, entering a blade passage at one radius and velocity, exiting in a similar manner, then the summation of moments due to forces acting on the control surface due to body forces acting on the fluid inside the control volume is

$$\Sigma T = \dot{m}(r_1 V_{\theta 1} - r_2 V_{\theta 2}) \tag{16}$$

where θ refers to the tangential component of fluid velocity.

Euler Turbine Equation

The basis for relating fluid flow to energy transfer is the Euler turbine equation, which can be derived from the preceding moment of momentum equation. The ideal energy transfer from the fluid is

$$P = \dot{m}w = \omega T \tag{17}$$

where T is now taken to be the resultant torque. Thus,

$$P = \dot{m}\omega(r_1 V_{\theta 1} - r_2 V_{\theta 2}) \tag{18}$$

$$\frac{P}{\dot{m}} = w_{ie} = \omega r_1 V_{\theta 1} - \omega r_2 V_{\theta 2} \tag{19}$$

Since the blade speed U is

$$U = r\omega \tag{20}$$

the equation may be written as

$$w_{ie} = U_1 V_{\theta 1} - U_2 V_{\theta 2} \tag{21}$$

Since $U_1 = U_2$ for an axial-machine, the equation may be written as

$$w_{ie} = U(V_{\theta 1} - V_{\theta 2}) \tag{22}$$

Efficiency

The efficiency of a turbomachine is a comparison of the actual work input or output with the work for a reversible process between the same initial state and final pressure. For a compressor or fan, the general expression for efficiency η is

$$\eta = \frac{w_{rev}}{w_{actual}} \tag{23}$$

For incompressible flow, the efficiency is

$$\eta = \frac{Hg}{w} = \frac{\rho Q Hg}{P} \tag{24}$$

where w is the work added to the fan, and P is the associated power.

Equation of State

For an ideal gas, the equation of state is

$$\rho = \frac{P}{RT} \tag{25}$$

Initial Efficiency Estimation

Moses [21] and Shepherd [27] both recommend the use of correlations of cascade test results based on the NASA publication SP-36 of Johnsen and Bullock [13] to estimate the preliminary efficiency at the meanline. The following development is typical for total pressure losses.

Total Pressure Loss

The total pressure loss in a fan blade row can be attributed (in this development) to three causes:

- profile losses due to boundary layers on the airfoil
- annulus losses due to boundary layers on the hub and casing
- secondary flow losses accounting for all other losses

Efficiency

The rotor efficiency may be approximated by

$$\eta = 1 - \frac{W_1^2 \bar{\omega}}{2(-w)} \tag{26}$$

where W_1 is the inlet fluid velocity relative to the blade, and w is the work per unit mass. The total pressure loss coefficient $\bar{\omega}$ may be expressed as a sum of three other loss coefficients described below:

$$\bar{\omega} = \bar{\omega}_p + \bar{\omega}_a + \bar{\omega}_s \tag{27}$$

Profile Loss Coefficient: The profile loss coefficient $\bar{\omega}_p$ may be directly derived in cascade experiments by measuring the boundary layers on the blade. This coefficient is directly related to the wake momentum thickness and may be expressed as

$$\bar{\omega}_p = 2\left(\frac{\theta_2}{c}\right)\left(\frac{c}{s}\right)\frac{\cos^2\beta_1}{\cos^3\beta_2} \tag{28}$$

Angles β_1 and β_2 refer to the inlet and exit relative flow angles respectively, while s/c is the blade solidity. The wake momentum thickness θ_2/c may be expressed (following experimental results) as:

$$\frac{\theta_2}{c} = \frac{0.004}{1 - 1.17 \ln(\frac{W_{max}}{W_2})} \tag{29}$$

The diffusion ratio $\frac{W_{max}}{W_2}$ depends on the turning angle $(\beta_2 - \beta_1)$ and on the spacing of blades. Based on experimental results, the diffusion ratio may be calculated using

$$\frac{W_{max}}{W_2} = \frac{\cos\beta_2}{\cos\beta_1}[1.12 + 0.61(\frac{s}{c})\cos^2\beta_1(\tan\beta_1 - \tan\beta_2)] \tag{30}$$

Annulus Loss Coefficient: While this loss cannot be measured in cascade tests, it can be estimated. A typical approximation is

$$\bar{\omega}_a = 0.02(\frac{c}{H})\frac{\cos^2\beta_1}{\cos^3\beta_m} \tag{31}$$

where the mean blade angle β_m is defined as

$$\tan\beta_m = 0.5(\tan\beta_1 - \tan\beta_2) \tag{32}$$

Secondary Loss Coefficient: Like the annulus loss coefficient, the secondary loss coefficient cannot be directly measured. However, it may be approximated as:

$$\bar{\omega}_s = 0.08\frac{\cos^2\beta_1}{\cos\beta_m}(\frac{s}{c})(\tan\beta_1 - \tan\beta_2)^2 \tag{33}$$

The mean blade angle β_m is defined above.

Deviation

In addition to the efficiency calculation, other correlations may be used to evaluate the meanline design. Cramer's rule is a correlation for the deviation angle δ:

$$\delta = m(\beta'_1 - \beta'_2)(\frac{s}{c})^{(1/2)} \tag{34}$$

where

$$m = (0.23)(2(\frac{a}{c})^2) + (\frac{\beta_2}{500}) \tag{35}$$

The angles in these equations are in degrees. The quantity (a/c) is the ratio of the distance between the leading edge and the point of maximum camber to the chord. For circular-arc camberlines, (a/c) = 0.5, and for parabolic camberlines, (a/c) = 0.25.

Tangent Difference Rule: In order to provide a safe surge margin, the design turning angle ($\beta_2 - \beta_1$) should be less than the stalling value. This results in an upper limit for the turning angle. The tangent difference rule recommends that the quantity ($\tan \beta_1 - \tan \beta_2$) be kept below a prescribed value (dependent on the solidity). The tangent difference rule is:

$$(\tan \beta_1 - \tan \beta_2)_{design} \leq \frac{1.55}{1 + 1.55(\frac{s}{c})} \tag{36}$$

Meanline Design Evaluation

At this point, a designer has two choices based on the meanline design, its efficiency, and associated performance correlations. If the evaluation perhaps indicates an unacceptable efficiency or turning angle, the designer may choose to change some of the performance specifications (like the pressure

rise or mass flow) and repeat the meanline design. If the meanline design appears acceptable, the designer may proceed to a more detailed design, as presented in the next section.

Three-Dimensional Blade Design

Introduction

The previous section discussed the meanline design of an axial-fan blade based on the assumption of two-dimensional flow. In many cases, the radial (spanwise) velocity is too large for the flow to be considered purely two-dimensional. Three-dimensional effects become more important for longer blades (small hub-to-tip ratios). Dixon [8] reports that radial flow should be included for hub to tip ratios ($\frac{r_i}{r_o}$) less than 4/5. For turbomachines with long blades ($\frac{r_i}{r_o} \leq 4/5$), the blade tip velocity is much larger than the blade hub velocity, and the blade is usually twisted. As a result, the meanline might not be representative of the entire blade, and in this case, the blade geometry and flow deflection should be described as a function of blade radius.

Radial Equilibrium

In an axial-flow turbomachine, flow is primarily in two directions--parallel to the axis of rotation (axial) and tangential to the blade cross-sections. In addition to these two flow directions, flow in the radial direction is often too large to be neglected. These radial flows are caused by an imbalance between the strong centrifugal forces pulling the fluid toward the blade tip and the radial pressure forces restoring equilibrium. Consequently, radial flow in an axial-flow turbomachine alters the mass flow distribution across the blade, thus changing the velocity distribution at the rotor outlet.

Radial equilbrium flow is defined as an axisymmetric flow in an annular passage with no radial velocity component. The flow streamlines lie on circular, cylindrical surfaces. A technique called the *radial equilibrium method* is widely used to consider three-dimensional effects in axial-flow

turbomachinery. This method is based on the assumption that any radial flow occurs within the blade row itself so that flow outside the row is in radial equilibrium, as illustrated in Figure 7.

Assuming incompressible flow with no gravitational effect and no initial radial motion, the radial pressure gradient due to the centrifugal effect may be expressed in terms of the tangential velocity and radius as:

$$\frac{dP}{dr} = \frac{\rho V_\theta^2}{r} \tag{37}$$

If the variation of tangential velocity with radius is known, then equation 37 provides an expression for the radial pressure gradient. Conversely, if a certain pressure gradient is imposed, then the variation of V_θ with r may be determined. In designing turbomachine blades, a designer usually chooses a tangential velocity distribution. Then equation 37 describes the radial pressure gradient necessary to maintain radial equilibrium. Several different relationships between the tangential velocity and radius will be considered in the following section.

Tangential Velocity Distribution

Numerous relationships between the tangential velocity V_θ and radial position are currently used. The following are most common.

Free Vortex Flow

The free vortex relationship stems directly from Newton's Law of Motion: in the absence of external forces, the angular momentum or moment of momentum is constant. In other words, in free

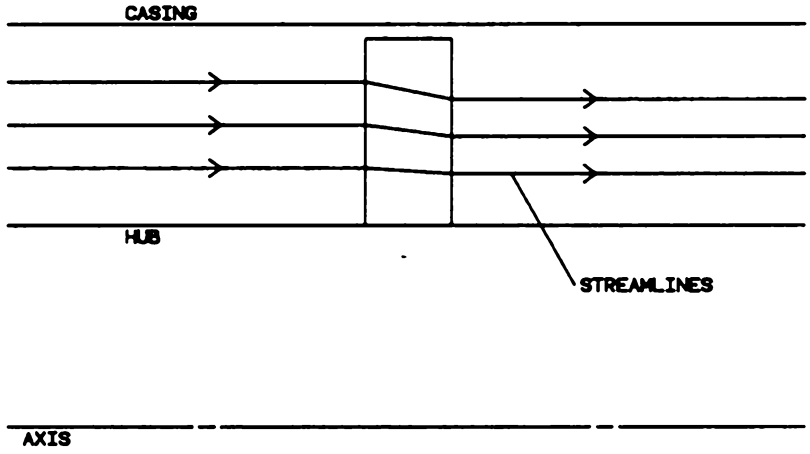

Figure 7. Flow Streamlines for Radial Equilibrium

vortex flow, the tangential velocity V_θ varies inversely with the radial position on the blade. That is,

$$V_\theta r = constant \tag{38}$$

The combination of the Euler equation, the pressure gradient equation, and the free vortex condition yields the conclusion that in an axial-flow turbomachine with free-vortex flow, the axial velocity is constant from blade hub to blade tip,

$$V_x = constant \tag{39}$$

A useful feature of free vortex flow is its easy adaptation to a other standard design criteria of equal energy transfer or work at all radii. The energy transfer in a turbomachine is proportional to the product of blade speed and tangential velocity (UV_θ), but the blade speed U is directly proportional to the radius while the tangential velocity V_θ is inversely proportional to the radius. Thus the energy transfer is constant and free vortex flow automatically assures equal energy transfer or work at all radii.

Forced Vortex Flow

Free vortex flow is a special case of general vortex flow which follows the relation

$$V_\theta r^n = constant \tag{40}$$

Although n = 1 holds for free vortex flow, n may have any value. In the case of forced vortex flow, n = -1. Forced vortex flow is often called solid rotational flow because the rotational motion of the fluid is that of a solid body with the tangential velocity V_θ directly proportional to the radial position r. Hence,

$$\frac{V_\theta}{r} = constant = \omega \tag{41}$$

In this case, the axial velocity V_x is not constant. Instead it varies in a complex manner from the blade hub to tip, being greater at the hub. Forced vortex flow is limited by the range of permissible hub ratios so that the condition for radial equilibrium does not prescribe negative axial velocities at the blade tip.

Other Methods of Tangential Velocity Distribution

Table 3, based on Horlock [12], summarizes tangetial velocity distributions. In addition it provides an overview of other common tangential velocity distribution methods.

The *constant reaction* method specifies tangential inlet and exit velocities so that the reaction is approximately the same at all radii:

$$V_{\theta 1} = ar - \frac{b}{r} \tag{42}$$

$$V_{\theta 2} = ar + \frac{b}{r} \tag{43}$$

where a and b are constants. The distribution of work is constant at all radii, but the condition of radial equilibrium is ignored.

The *half vortex* method specifies a tangential velocity that is the arithmetic mean of the free vortex and constant reaction tangential velocity distributions. In addition, the axial velocity is constant at all radial positions, like free vortex flow, but radial equilibrium is neglected.

The *constant* α_2 method includes constant work distribution, constant axial velocity, and constant inlet angle to the stator (α_2), but no radial equilibrium.

Table 3. Tangential Velocity Distributions

	Work Variation w/ Radius	Tangential Velocity Distribution	Axial Velocity Distribution	Radial Equilibrium
Free Vortex	constant	$V_\theta r = c$	constant	yes
Constant Reaction (a)	supposed constant	$V_\theta = ar \pm b/r$	supposed constant	ignored
Half Vortex	supposed constant	averaged	supposed constant	ignored
Constant α_2	supposed constant	$V_\theta = a - b/r$	supposed constant	ignored
Constant Reaction (b)	constant	$V_\theta = ar \pm b/r$	from radial equilibrium	yes
Forced Vortex	increases w/ radius	$V_\theta \propto r$	from radial equilibrium	yes
Exponential	constant	$V_\theta = a \pm b/r$	from radial euilibrium	yes

A second type of *constant reaction* method combines the previous conditions of constant reaction with radial equilibrium.

Finally, the *exponential* method dictates that the tangential velocity have the form

$$V_\theta = a \pm \frac{b}{r} \tag{44}$$

It provides equal work distribution, constant inlet angle to stator (α_2), and radial equilibrium, but the axial velocity decreases as radius increases.

Although each of these methods has its own advantages, the free vortex method is most commonly selected for its simplicity and ease of application. All subsequent calculations in this thesis are based on free vortex flow. For example, a fan designer might extend his meanline design by applying free vortex flow conditions. The result is discrete values of inlet and exit blade and flow angles as a function of radius. The radial positions on the blade at which these calculations are performed are selected so that equal mass flows between any two adjacent sections. By applying free-vortex flow, this also means that equal work is being done between any two adjacent sections.

The next step in designing a three-dimensional blade is the decision of what goes between the inlet and exit angles--the camberline.

Blade Cross Sections

Many different shapes are used as cross sections for axial-flow fans. In cases where the speed is low and the concern for efficiency is not great, flat plates often suffice. However, the introduction of curvature or camber usually increases the lift capabilities and strength of the blade, and often aircraft airfoil sections are used for fan blading. An *airfoil* may be geometrically defined as a meanline or camberline on which a certain thickness form is superimposed.

Camberline Shapes

The camberline may be simply regarded as the connecting line between the rotor inlet blade angle β'_1 and the rotor outlet blade angle β'_2. However, the camberline plays a significant role in the cross section because it determines the lifting and deflection properties of the blade element.

The camberline is often formed of one or more circular arcs or of one or two parabolic arcs. Often a single arc is used for geometric simplicity, and all further developments in this thesis will concentrate on camberlines composed of a single circular arc.

Two important parameters are the camber angle θ and the position of maximum camber a. The camber angle θ represents the total angle through which the camberline is bent; thus it also represents the total angle of fluid deflection. The camber angle is therefore the difference between the inlet and outlet blade angles:

$$\theta = \beta'_1 - \beta'_2 \tag{45}$$

The position of maximum camber a is usually expressed as a percentage of the chord c, as a/c. For a given airfoil thickness distribution along the chord, as the point of maximum camber approaches the leading edge, the largest surface velocities occur closer to the leading edge. The result is improvement in operating range, low-speed performance, and delay of stall, but a susceptibility to cavitation in pumps. In contrast, if the point of maximum camber lies closer to the trailing edge, the operating range is narrower, the stalling angle of attack is lower, but the cavitation behavior is better. In a circular arc camberline, the point of maximum camber is fixed at fifty percent of the chord.

Figure 8 illustrates a typical circular-arc camberline. The camber angle θ and maximum camber position a/c are noted. By specifying the type of camberline, the designer can improve the pictorial representation of the blade. Previously, only inlet and exit velocity triangles were known, Now, the

camberline adds more information regarding what happens between the inlet and exit of a blade passage, as shown in Figure 9.

Thickness Distribution

Many blade profiles are formed by bending a symmetrical airfoil section on the curved camberline. A typical (NACA 65-series) base profile, shown in Figure 10, is divided into stations along the x-axis, as percentages of the length L. At each station the height of the profile (also as a percentage of the length L) is specified. Often the leading edge is a circular arc blended into the main profile. Ideally, the trailing edge should be sharp, but strength and manufacturing requirements usually result in a rounded trailing edge.

Two important parameters of the base profile are the maximum thickness and its position on the cross section. The maximum thickness is usually expressed as a percentage of the chord, and should be kept as small as possible. Turton [30] recommends that the maximum thickness be less than twelve and a half percent of the chord. Sometimes strength considerations force larger thicknesses, but the reduction in aerodynamic performance can be drastic. Moving the position of maximum thickness has the same effect on performance as the previously discussed position of maximum camber.

Many airfoil sections are available to the turbomachine designer. An excellent source of profiles is Abbott [1]. The NACA 65-series airfoils were originally developed for high-Mach number applications in compressors, but are also widely used in lower-speed fans. The thickness distribution for the standard ten-percent maximum thickness NACA 65-series sections appears in Appendix A. Other thickness forms of the 65-series are obtained by linearly adjusting the ten-percent coordinates. The program FAN3D in this thesis uses NACA 65-series sections to draw fan blades, but is easily modified to include other profile sections.

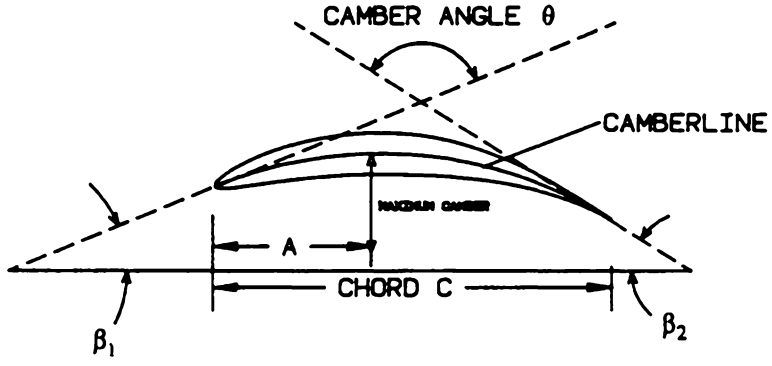

Figure 8. A Typical Cascade Camberline

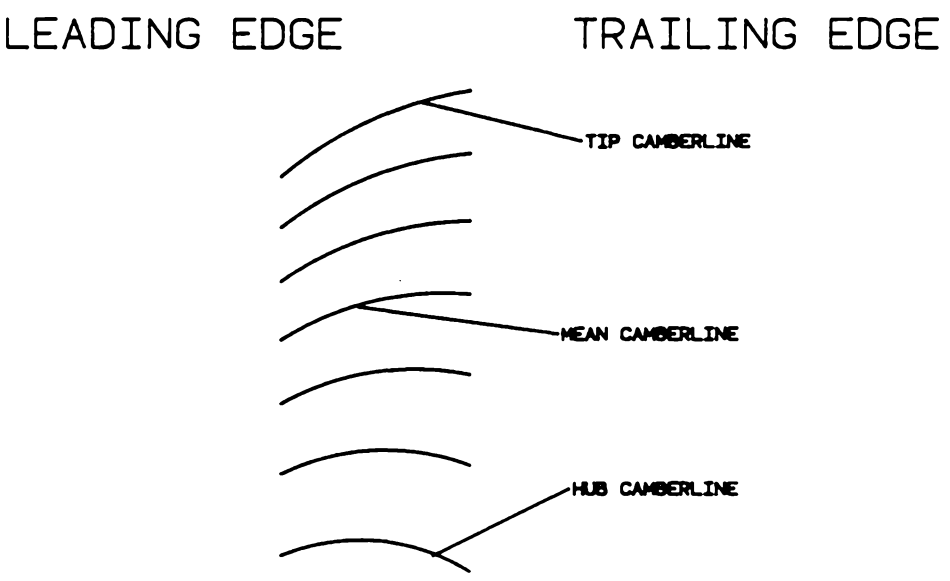

Figure 9. Typical Camberlines as a Function of Blade Radius

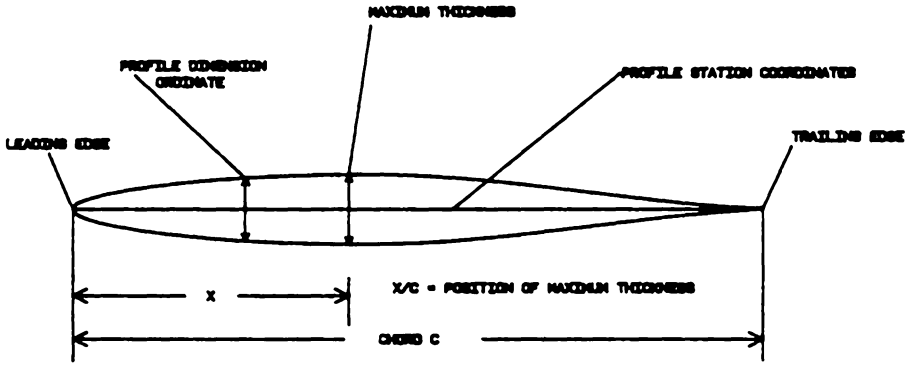

Figure 10. Typical Base Profile

The introduction of thickness to the blade sections again improves the information available to the designer. Figure 9 has been modified to include thickness as well as camberlines, as shown in Figure 11.

Stacking the Blade Cross Sections

The final step in designing a three-dimensional blade is the radial stacking or positioning of the blade cross sections. This eliminates the artificial aspect of Figure 11 in which cross sections are laid out side-by-side instead of above one another. Stress, geometric, and aerodynamic considerations affect the optimum orientation of the cross sections. Both Balje [2] and Wallis [33] recommend that the centroids of the cross sections be constrained to lie on a radial line, thus relieving blade-root stresses. The result is a three-dimensional blade, as shown in Figure 12. A designer is now ready to reevaluate the performance of the turbomachine based on a three-dimensional blade design.

Performance Evaluation

With a three-dimensional model of an axial-flow fan, a reevaluation of its performance is in order. Previously, the efficiency was calculated based on the basis of cascade correlations applied at the blade's mean radius. Ideally, the known variation of blade geometry in the radial direction should now allow better calculations of efficiency. However, most sources do not include these effects. In fact, most sources (Csanady [7], Dixon [8], Eck [9], Horlock [12], Logan [17], Oborne [23], Shepherd [27], and Wallis [32]) rely only on cascade correlations like those presented for the meanline design in this thesis. These sources attribute total pressure losses in an axial-flow turbomachine to three sources: profile losses, annulus and endwall losses, and secondary losses. The losses are calculated using experimental cascade correlations, and when differing terminologies

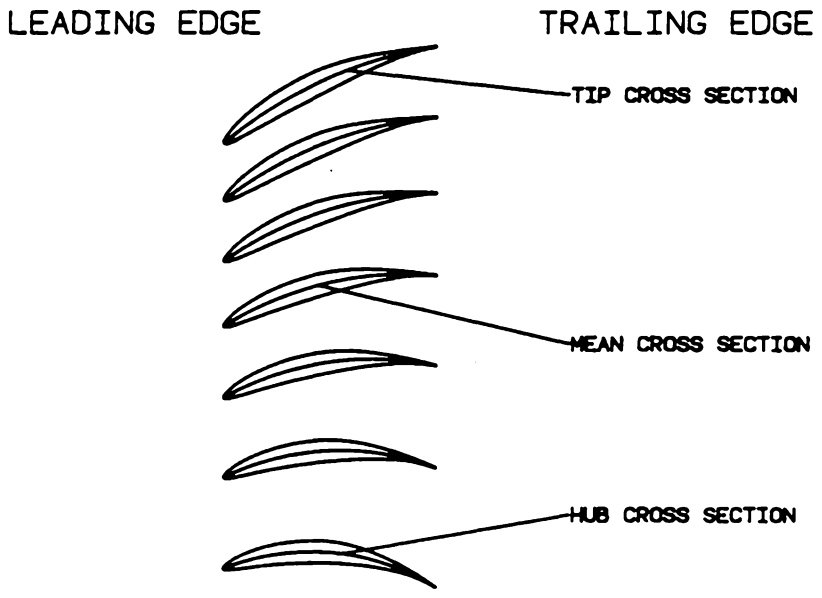

Figure 11. Blade Cross Sections as a Function of Radius

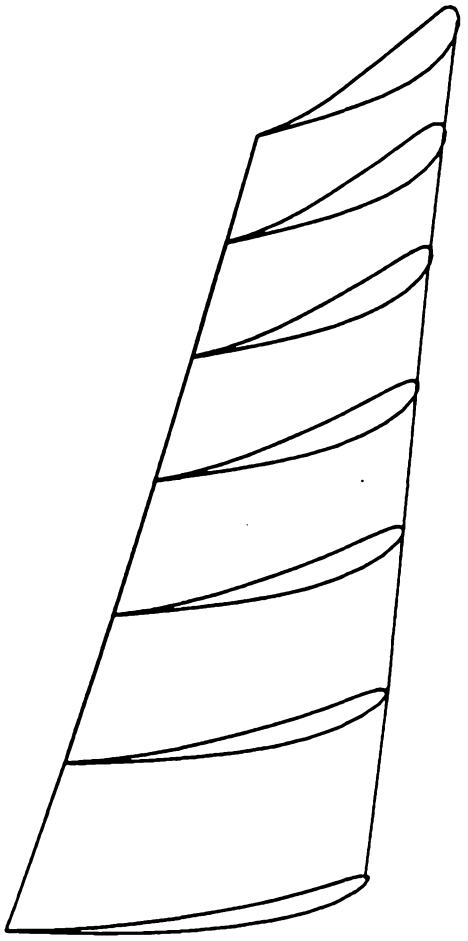

Figure 12. A Three-Dimensional Representation of a Fan Blade

are reduced to a consistent set of expressions for losses, the primary source usually is the NASA publication SP-36 by Johnsen and Bullock [13].

Balje [2] is the only reference which discusses efficiency calculations explicitly considering the radial geometry variations in the blade. Balje notes that sometimes it *is* sufficient to calculate flow path performance by considering only the meanline. Often this is the case for blades with comparatively high hub to tip ratios (short blades) but not necessarily for longer blades.

For longer blades, Balje recommends that the loss evaluation be made at three different blade radii--tip, mean, and hub radii-- and then averaged to determine the overall efficiency. This method is especially useful for blades designed using free vortex flow (as those in this thesis are). In order to maintain equal energy transfer close to the rotor hub, free vortex flow causes high flow deflection. This results in a highly twisted blade with possibly large losses in the hub region. Balje's averaging loss method permits the effect of high hub losses to be directly investigated.

Of course, the cascade correlations in the meanline design or any of those cited by the other listed sources could also be averaged in this manner. The following expressions are Balje's recommended loss coefficients, presented using the nomenclature established in this thesis.

Efficiency

The term that Balje [2] refers to as hydraulic efficiency is

$$\eta = 1 - \frac{\Phi^2 \Sigma\zeta}{2\psi} \tag{46}$$

where

$$\Sigma\zeta = \Sigma\text{overall losses} = \zeta_p + \zeta_a + \zeta_c$$

Φ = flow factor

ψ = head coefficient

The head coefficient ψ is defined as

$$\psi = \frac{gH}{\omega^2 D^2} \tag{47}$$

And the flow factor Φ is

$$\Phi = \frac{8}{\pi[1 - (\frac{r_i}{r_o})^2] n_s d_s} \tag{48}$$

The loss coefficients ζ_p, ζ_e, and ζ_c are defined below.

Losses

Balje [2] also bases losses on the three sources: profile, endwall, and clearance losses. Before these losses are presented the following parameters should be defined.

Blade Loading Factor: At radial position r_x, the blade loading factor is

$$\delta u_x = \frac{\pi d_s[1 - (\frac{r_i}{r_o})^2]}{(\frac{r_x}{r_o})\eta n_s} \tag{49}$$

Mean Vector Angle: At radial position r_x, the mean vector angle is

$$\cot \lambda_\infty = \pi d_s[1 - (\frac{r_i}{r_o})^2][\frac{1}{4(\frac{r_x}{r_o})\eta n_s} - \frac{n_s d_s^2}{8(\frac{r_o}{r_x})}] \tag{50}$$

Circulation Parameter

$$C = \frac{2\delta u}{(s/c)\{1 + [\cot \lambda_\infty - (\delta \frac{u}{2})^2]\}} \tag{51}$$

Equivalent Diffusion Factor (empirical relation)

$$D = 1 + 0.9(\frac{a}{c}) + 0.3C \tag{52}$$

Momentum Thickness (empirical relation)

$$\frac{\theta}{l} = \frac{0.013}{2.63 - D} - 0.004 \tag{53}$$

Profile Loss Coefficient

$$\zeta_p = \frac{1.286H^2(\frac{s}{c})(\frac{\theta}{l})[1 + \cot \lambda_\infty + (\frac{\delta u}{2})^2]^{1.5}}{[1 - H(\frac{s}{c})(\frac{\theta}{l})\sqrt{1 + \cot \lambda_\infty + (\frac{\delta u}{2})^2}]^3} \tag{54}$$

where

$\frac{s}{c}$ = solidity

$\frac{\theta}{l}$ = total momentum thickness

λ_∞ = mean vector angle

H = blade shape factor

δu = blade loading factor

Endwall Loss

$$\zeta_e = \zeta_p\left(\frac{A_e}{A_p}\right) \tag{55}$$

where $\frac{A_e}{A_p}$ is the ratio of endwall frictional area to profile frictional area.

Clearance Loss

$$\zeta_c = \frac{0.19\delta u\left[\cot\lambda_\infty + \left(\frac{\delta u}{2}\right)^2\right]}{\sqrt{1 + \cot^2\lambda_\infty}}\left[\left(\frac{H}{g}\right)\left(\frac{H}{c}\right)\right]^{(-0.5)} \tag{56}$$

where H/g is the ratio of blade height to clearance gap.

Three-Dimensional Evaluation

At this point a designer has a number of choices based on the three-dimensional blade design and its efficiency evaluation. If the design is not acceptable, one choice is to change some performance specifications (like pressure rise or mass flow rate) and return to the meanline design. Otherwise, an unacceptable design might be improved by revising the three-dimensional design with a different tangential velocity distribution or perhaps with a different camberline, section profile, or stacking scheme. Assuming that the designer is satisfied with the current design, the next topic is mechanical strength of an axial-flow fan blade.

Mechanical Strength Considerations

Sometime during the design of a turbomachine, aerodynamic concerns must be laid aside in order to investigate the mechanical strength of the blading. Often these two concerns are diametrically opposed. For example, from an aerodynamic point of view, a blade should be as thin as possible, with a sharp trailing edge, but from a mechanical strength viewpoint, the same blade should be thick with a rounded trailing edge. While these two concerns are not completely independent, a blade is often first aerodynamically designed without great concern for mechanical strength (as was done in the previous sections). Then, the design is checked for mechanical strength. Depending on the conclusions drawn from this check, a designer may change the fan blade material, revise the three-dimensional blade design, or change performance specifications (like the pressure rise per stage) and begin a new design at the meanline.

An excellent development of mechanical strength calculations for turbomachine blading is presented by Fielding [10]. The basic goal of Fielding is to present finished program codes for various rotating equipment design calculations for use on a programmable TI-59 calculator. However, Fielding's development and numerical computational presentations can be easily extended to whatever computing facilities are available to the designer.

Blade Section Properties

The cross-sectional properties at various positions along the blade length are often required in both aerodynamic and mechanical calculations. The following development discusses the calculation of the following cross-sectional properties: area, position of center of gravity, the maximum and minimum moments of inertia, and the angular position of the minimum moment of inertia relative to the cross-sectional axes.

Input Data

In order to calculate the blade cross section properties, the density of the blading material and the cross section geometry must be known. Fielding recommends that the geometry be described in cartesian coordinates with the origin placed in the estimated vicinity of the section's center of gravity. Figure 13 illustrates a typical cross section and its coordinate system. The section is divided into n strips and at each station along the x-axis ($x_1, x_2, \ldots, x_n$) the ordinates of both the upper and lower surfaces ($y_{U1}, y_{U2}, \ldots, y_{Un}$ and $y_{L1}, y_{L2}, \ldots, y_{Ln}$) should be known. Care should be taken to assign each coordinate the proper (positive or negative) sign relative to the chosen axis.

Nomenclature

The following is a summary of the nomenclature used in this section.

A	area
F	product of inertia about the X and Y axes
F_0	product of inertia about a set of axes parallel to the X and Y axes, through the centroid
I_{xx}	moment of inertia about the X axis
I_{yy}	moment of inertia about the Y axis
$(I_{xx})_0$	moment of inertia about an axis parallel to the X axis through the section centroid
$(I_{yy})_0$	moment of inertia about an axis parallel to the Y axis through the section centroid
I_{max}	section maximum moment of inertia
I_{min}	section minimum moment of inertia
θ	inclination of the minimum moment of inertia axis relative to the chosen axes

n	number of strips into which the section is divided
$\overline{X}$	distance of section centroid from the Y axis
$\overline{Y}$	distance of section centroid from the X axis
ρ	blade material density

Calculations

Using these symbols, the cross section properties may be calculated in the following manner.

The area of a given cross section is

$$A = \sum_{i=1}^{n} \frac{X_{i+1} - X_i}{2}(Y_{U,i} + Y_{U,i+1} - Y_{L,i} - Y_{L,i+1}) \tag{57}$$

The distance of the section centroid from the Y axis is

$$\overline{X} = \sum_{i=1}^{n} \frac{(X_{i+1}^2 - X_i^2)(Y_{U,i} + Y_{U,i+1} - Y_{L,i} - Y_{L,i+1})}{4A} \tag{58}$$

The distance of the section centroid from the X axis is

$$\overline{Y} = \sum_{i=1}^{n} \frac{(X_{i+1} - X_i)[(Y_{U,i} + Y_{U,i+1})^2 - (Y_{L,i} + Y_{L,i+1})^2]}{8A} \tag{59}$$

The product of inertia about the X and Y axes is

$$F = \sum_{i=1}^{n} \frac{X_{i+1}^2 - X_i^2}{16}[(Y_{U,i} + Y_{U,i+1})^2 - (Y_{L,i} + Y_{L,i+1})^2] \tag{60}$$

The moments of inertia about the X and Y axes respectively are

$$I_{xx} = \sum_{i=1}^{n} \frac{X_{i+1} - X_i}{24}[(Y_{U,i} + Y_{U,i+1})^3 - (Y_{L,i} + Y_{L,i+1})^3] \tag{61}$$

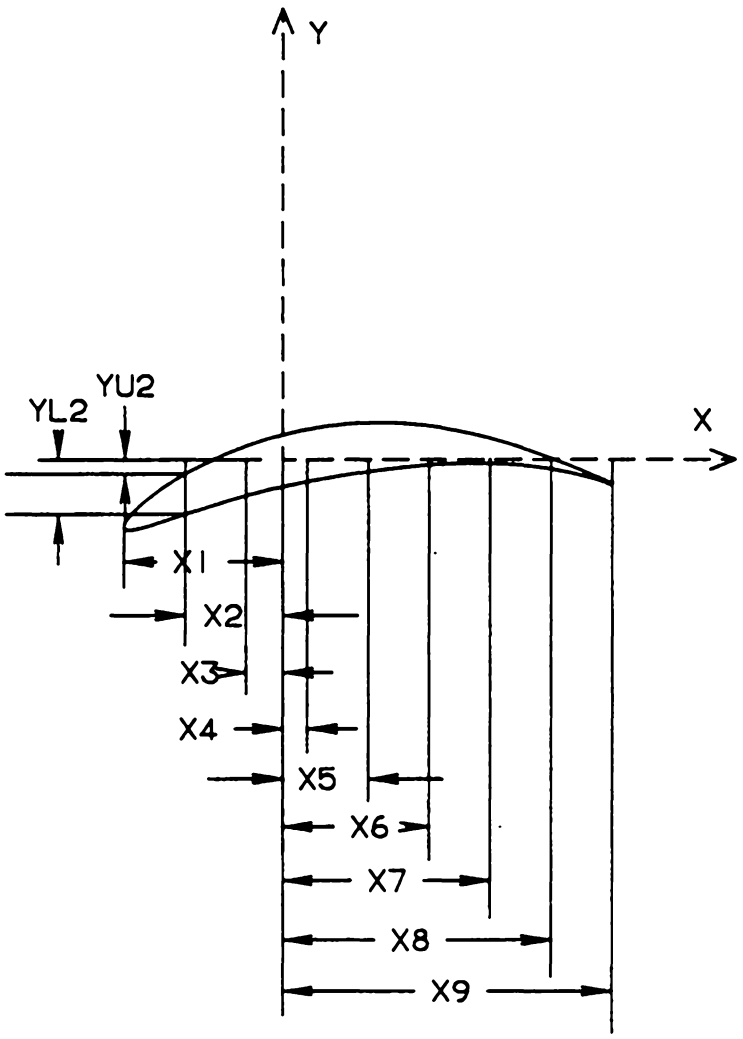

Figure 13. Cartesian Coordinates for Blade Cross Section.

$$I_{yy} = \sum_{i=1}^{n} (X_i - X_{i+1})(Y_{U,i} + Y_{U,i+1} - Y_{L,i} - Y_{L,i+1})[\frac{(X_i + X_{i+1})^2}{8} + \frac{(X_i - X_{i+1})^2}{24}] \quad (62)$$

These section properties are calculated with respect to the axes shown in Figure 14. The parallel axis theorem allows the properties to be transferred to parallel axes passing through the section centroid.

$$(I_{xx})_0 = I_{xx} - A\overline{Y}^2 \quad (63)$$

$$(I_{yy})_0 = I_{yy} - A\overline{X}^2 \quad (64)$$

$$F_0 = F - \overline{X}\,\overline{Y}A \quad (65)$$

Then, the maximum and minimum moments of inertia and the inclination of the axis of minimum moment of inertia are:

$$I_{max} = \frac{1}{2}\{((I_{yy})_0 + (I_{xx})_0 + [(I_{yy})_0 - (I_{xx})_0](\sec 2\theta)\} \quad (66)$$

$$I_{min} = \frac{1}{2}\{((I_{yy})_0 + (I_{xx})_0 - [(I_{yy})_0 - (I_{xx})_0](\sec 2\theta)\} \quad (67)$$

$$\tan 2\theta = \frac{2F_0}{(I_{yy})_0 - (I_{xx})_0} \quad (68)$$

Centrifugal Stresses

The evaluation of stresses in turbomachine blading is an involved process, usually requiring large computer algorithms to account for the complex blade geometry. For a first approximation, the calculations are often simplified and the results are sufficiently accurate to check the mechanical

strength of blades at the initial design stages. The following procedure for calculating centrifugal stresses in a turbomachine blade assumes that the following items are known:

- blade cross-sectional area as a function of radius
- the blade material density
- the speed of rotation

The blade may be shrouded or unshrouded. If it is shrouded, then the shroud centrifugal force must also be known in order to account for its effect on the centrifugal blade stresses. It should be noted that the stress at the blade tip is zero for an unshrouded blade, but is nonzero for a shrouded blade.

Nomenclature

The following symbols are used in the calculations of centrifugal stresses:

A_r	blade cross-sectional area at radius r
A	blade cross-sectional area
F_c	centrifugal force
F_s	shroud centrifugal force
g	gravitational acceleration constant
n	number of radii for which an area is specified
r	radius
r_i	hub radius
r_o	tip radius
ρ	blade material density
σ	stress

σ_r	stress at radius r
ω	rotational speed

Equations

Referring to Figure 14, the centrifugal force of an element of thickness *dr* is

$$dFc = \rho\omega^2 Ardr \tag{69}$$

Then the associated centrifugal force above radius r is

$$F_c = \rho\omega^2 \int_r^{r_o} Ardr \tag{70}$$

and the centrifugal stress on section r is

$$\sigma_r = \frac{\rho\omega^2}{Ar} \int_r^{r_1} Ardr \tag{71}$$

The equivalent for numerical calculations (of equation 71) is

$$\sigma_r = \frac{\rho\omega^2}{Ar} \sum_{i=1}^{r} \frac{A_i r_i + A_{i+1} r_{i+1}}{2} (r_i - r_{i+1}) \tag{72}$$

Thus, equation 72 allows the calculation of the centrifugal stresses in a turbomachine blade at radial positions along the blade length where cross-sectional areas are known.

Offset Bending Stresses

Another important consideration in the evaluation of mechanical strength in turbomachine blading is the magnitude of bending stresses. As in the case of centrifugal stressses, an exact analysis re-

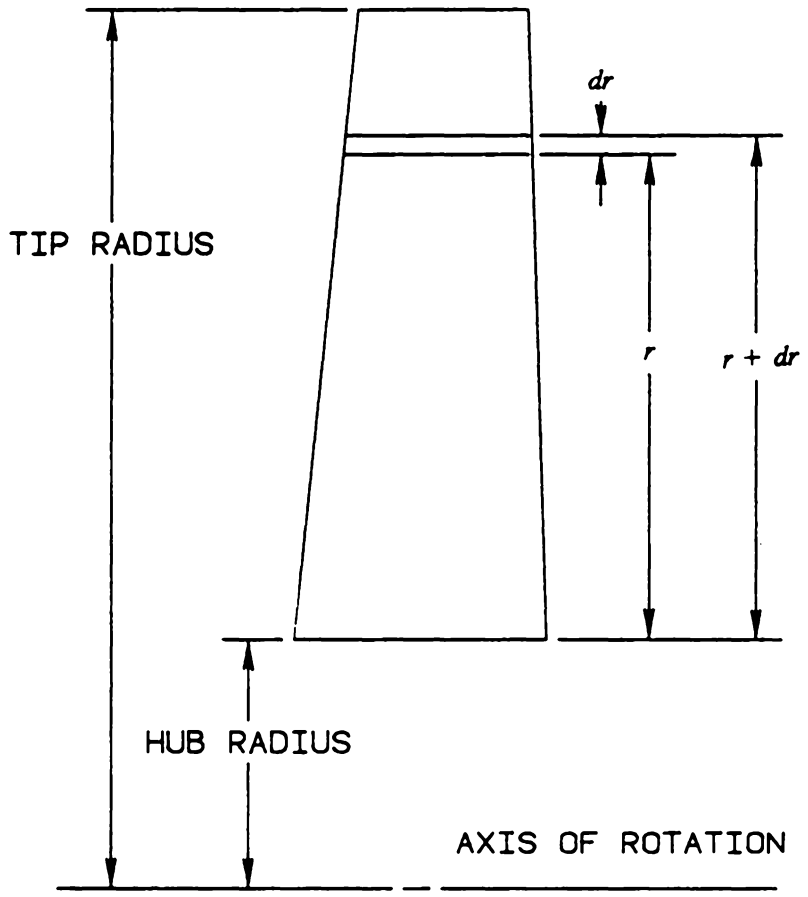

Figure 14. Centrifugal Force on a Differential Element

quires complex treatment. However, a simplified approach is often sufficient in the early evaluation of mechanical strength. Fielding [10] analyzes bending stresses which result when the centers of gravity of the blade cross sections are offset from a radial line passing through the root cross-sectional center of gravity. While the previous chapter recommended that the cross sections be stacked so that all centers of gravity lie on a radial line, manufacturing constraints may not permit this. In this case, the reader is referred to the development for the calculation of offset bending stresses in Fielding [10].

Other forms of bending stresses exist in turbomachine blades in addition to the bending stresses that result from the pressure differential across the blade. Even in a blade whose cross section centroids are aligned with the root centroid, the blade twist prevents the trailing edge from lying in a radial line, thus causing some bending stresses.

Geometric Modeling

Current Blade Design

Based on the previous chapters on two- and three-dimensional aerodynamic design and mechanical strength, the basic design of a simple axial-flow fan blade might be considered finished. But what kind of information has the designer determined? First, the overall dimensions of the blade have been set, such as the hub radius, hub-to-tip radius ratio, and the chord (which is a function of the radial position along the blade length). Additionally, the designer has determined other geometrical aspects at discrete radial positions along the blade length. These aspects are

- inlet and exit blade angles
- inlet and exit flow angles
- camberline shapes
- thickness distributions

All of these aspects collectively describe individual cross sections of the blade, but even these cross sections are described only in the simplest terms. For example, the thickness is represented by the points defined by the chosen thickness form (in this case the NACA 65-series), but at this stage, the designer does not have a method for smoothly connecting or interpolating these points. More importantly, the designer has no method of interpolating between two adjacent cross sections. Thus, if the inlet and exit blade angles at a place halfway between cross sections A and B must be known, then the designer must backtrack to the free vortex flow calculations.

What is Geometric Modeling?

An alternative to these additional calculations is *geometric modeling*. Mortenson [20] defines *geometric modeling* as a collection of computational methods whose goal is to define the mathematical shape and geometric characteristics of an object. The result is a model or representation of the original object, but the model is easier and more convenient to use and analyze than the original. The process of geometric modeling is closely tied to computers that efficiently perform the mathematics necessary to create a model, then store the model, and later analyze the model. Until now, the presentation in this thesis has not been heavily dependent on computers, but all subsequent discussions will depend on computers for their computational, analytical, and graphical support.

Three aspects of geometric modeling are representation, design, and rendering. They may be defined as follows.

- *Representation* is the task of developing a mathematical model of a given physical object. The shape is assumed fixed and the model is created only once.

- *Design* is the process of creating a new shape to meet given specifications. The geometric variables defining the model are modified until these specifications are met.

- *Rendering* is the task of producing a visual representation of a mathematical model.

All three aspects of geometric modeling will be discussed in detail in the following sections.

Parametric Equations

Mathematical equations or curves may be classified as either *parametric* or *nonparametric*. The nonparametric equation is usually more familiar, being of the explicit form

$$y = f(x) \tag{73}$$

or of the implicit form

$$f(x,y) = 0 \tag{74}$$

In a nonparametric equation, the dependent variable y is expressed as a function of the independent variable x. However, in the parametric form, both the independent and dependent variables are expressed in terms of a separate variable called a *parametric variable.* Mortenson [20] defines a parametric curve as a "point-bounded collection of points whose coordinates are given by continuous, one-parameter single-valued mathematical functions." A parametric equation has the form

$$x = x(u) \tag{75}$$

$$y = y(u) \tag{76}$$

$$z = z(u) \tag{77}$$

The parametric variable u is usually bounded by the interval $u \in [0,1]$, and the curve has a *positive sense* in the direction of increasing u. Often a parametric curve is expressed as a vector:

$$\mathbf{p} = [x(u) \;\; y(u) \;\; z(u)] \tag{78}$$

The tangent vector $\mathbf{p}^u(u)$ can be found by differentiating equation 78 so that

$$\mathbf{p}^u = \frac{d\mathbf{p}(u)}{du} \tag{79}$$

or

$$\mathbf{p}^u = \left[\frac{dx(u)}{du} \;\; \frac{dy(u)}{du} \;\; \frac{dz(u)}{du} \right] \tag{80}$$

The parametric derivative is related to the ordinary derivative by

$$\frac{dy}{dx} = \frac{(dy/du)}{(dx/du)} \tag{81}$$

An example of a parametric curve is

$$x(u) = a + lu \tag{82}$$

$$y(u) = b + mu \tag{83}$$

$$z(u) = c + nu \tag{84}$$

where a,b,c,l,m,n are all constants. Equations 82 through 84 represent a straight line whose endpoints are $p(0) = [a \;\; b \;\; c]$ and $p(1) = [a + l \;\; b + m \;\; c + n]$. Although parametric equations may seem more complicated and cumbersome than nonparametric equations, parametric equations are the cornerstone of geometric modeling. Some of their advantages include:

- The dependent and independent variables are separated, allowing greater insight into the control and behavior of the curves.

- Parametric equations usually have more degrees of freedom to control the shape of a curve than do nonparametric curves.

- Transformations like rotation or translation can be performed directly on the parametric equations.

- Parametric equations are inherently bounded by the parametric variable (usually $u \in [0,1]$).

- Parametric equations are independent of coordinate system, as are the shapes of most objects. Thus, parametric equations are well-suited to fitting a curve or surface through a group of points, regardless of the choice of coordinate system.

- Parametric equations can easily handle the infinite slopes of vertical tangent lines or planes. Since

$$\frac{dy}{dx} = \frac{(dy/du)}{(dx/du)} \tag{85}$$

then $dx/du = 0$ indicates that dy/dx has an infinite slope, if $dy/du \neq 0$.

- Finally, parametric equations often are the only method of representing a nonplanar, bounded curve.

Parametric Cubic Curves

The following development will concentrate on the use of parametric curves and surfaces to model turbomachine blades. The next decision concerns the order of the parametric polynomial. In general, a parametric curve in three dimensions has the form

$$\mathbf{p} = [x(u) \;\; y(u) \;\; z(u)] \tag{86}$$

where

$$x(u) = a_{0x} + a_{1x}u + a_{2x}u^2 + \cdots a_{nx}u^n \tag{87}$$

$$y(u) = a_{0y} + a_{1y}u + a_{2y}u^2 + \cdots a_{ny}u^n \tag{88}$$

$$z(u) = a_{0z} + a_{1z}u + a_{2z}u^2 + \cdots a_{nz}u^n \tag{89}$$

and $a_{0x}, a_{1x}, \ldots, a_{nx}; a_{0y}, a_{1y}, \ldots, a_{ny}; a_{0z}, a_{1z}, \ldots, a_{nz}$ are all constants.

If all the constants except a_{0x}, a_{0y}, a_{0z} are zero, then the parametric curve describes a point. Similarly, a parametric curve composed of only linear combinations of the parametric variable u represents a line.

The simplest parametric polynomial capable of representing a curve that can twist in space and can have points of inflection is a parametric cubic polynomial. It has a form like

$$x(u) = a_{0x} + a_{1x}u + a_{2x}u^2 + a_{3x}u^3 \tag{90}$$

$$y(u) = a_{0y} + a_{1y}u + a_{2y}u^2 + a_{3y}u^3 \tag{91}$$

$$z(u) = a_{0z} + a_{1z}u + a_{2z}u^2 + a_{3z}u^3 \tag{92}$$

The following section discusses the forms of parametric cubic (pc) curves and their application to modeling axial-flow fan blades.

Algebraic Form

The most commonly known form of a parametric cubic curve is the *algebraic form*:

$$x(u) = a_{0x} + a_{1x}u + a_{2x}u^2 + \cdots a_{3x}u^3 \tag{93}$$

$$y(u) = a_{0y} + a_{1y}u + a_{2y}u^2 + \cdots a_{3y}u^3 \tag{94}$$

$$z(u) = a_{0z} + a_{1z}u + a_{2z}u^2 + \cdots a_{3z}u^3 \tag{95}$$

where the parametric variable u is the independent variable, usually bounded by $u \in [0, 1]$.

The twelve constant coefficients in equations 93 through 95 are called the *algebraic coefficients*. These coefficients determine the size, shape, and position of the curve.

Often a parametric curve is expressed in a more compact vector form. Equations 93 through 95 may be written as

$$\mathbf{p} = \mathbf{a}_3u^3 + \mathbf{a}_2u^2 + \mathbf{a}_1u + \mathbf{a}_0 \tag{96}$$

where $\mathbf{p}(u)$ is a postion vector at any point on the parametric cubic curve, and the vectors $\mathbf{a}_3, \mathbf{a}_2, \mathbf{a}_1$, and $\mathbf{a}_0$ are composed of the corresponding scalar algebraic coefficients in equations 93 through 95.

Geometric Form

Although the algebraic form of a parametric cubic (pc) curve is most recognizable as a cubic parametric equation, it is not the most convenient form for most geometric modeling purposes. In its *geometric form*, a parametric cubic curve is described in terms of its endpoints--their position and tangent vectors. The result is a form which is easier to understand, visualize, and control.

The geometric form of a parametric cubic curve may be derived from the algebraic form by substituting the two endpoints $\mathbf{p}(0)$ and $\mathbf{p}(1)$ and their corresponding tangent vectors $\mathbf{p}^u(0)$ and $\mathbf{p}^u(1)$ in equations 93 through 95. This is accomplished by substituting $u = 0$ and $u = 1$ into the equations to find $\mathbf{p}(0)$ and $\mathbf{p}(1)$. Then, equations 93 through 95 can be differentiated with respect to u, and $\mathbf{p}^u(0)$ and $\mathbf{p}^u(1)$ can be evaluated by substituting $u = 0$ and $u = 1$ into these equations. Four equations result:

$$\mathbf{p}(0) = \mathbf{a}_0 \tag{97}$$

$$\mathbf{p}(1) = \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3 \tag{98}$$

$$\mathbf{p}^u(0) = \mathbf{a}_1 \tag{99}$$

$$\mathbf{p}^u(1) = \mathbf{a}_1 + 2\mathbf{a}_2 + 3\mathbf{a}_3 \tag{100}$$

Equations 97 through 100 may be solved so that the algebraic coefficients are expressed in terms of the boundary conditions $\mathbf{p}(0)$, $\mathbf{p}(1)$, $\mathbf{p}^u(0)$, and $\mathbf{p}^u(1)$.

$$\mathbf{a}_0 = \mathbf{p}(0) \tag{101}$$

$$\mathbf{a}_1 = \mathbf{p}^u(0) \tag{102}$$

$$\mathbf{a}_2 = -3\mathbf{p}(0) + 3\mathbf{p}(1) - 2\mathbf{p}^u(0) - \mathbf{p}^u(1) \tag{103}$$

$$\mathbf{a}_3 = 2\mathbf{p}(0) - 2\mathbf{p}(1) + \mathbf{p}^u(0) + \mathbf{p}^u(1) \tag{104}$$

Substitution of equations 101 through 104 in equation 96 results in an expression for the parametric cubic curve in terms of its boundary conditions:

$$\mathbf{p}(u) = (2u^3 - 3u^2 + 1)\mathbf{p}(0) + (-2u^3 + 3u^2)\mathbf{p}(1) +$$

$$(u^3 - 2u^2 + u)\mathbf{p}^u(0) + (u^3 - u^2)\mathbf{p}^u(1) \tag{105}$$

The geometric form can be further simplified by substituting

$$F_1(u) = 2u^3 - 3u^2 + 1 \tag{106}$$

$$F_2(u) = -2u^3 + 3u^3 \tag{107}$$

$$F_3(u) = u^3 - 2u^2 + u \tag{108}$$

$$F_4(u) = u^3 - u^2 \tag{109}$$

so that

$$\mathbf{p}(u) = F_1\mathbf{p}(0) + F_2\mathbf{p}(1) + F_3\mathbf{p}^u(0) + F_4\mathbf{p}^u(1) \tag{110}$$

The terms $\mathbf{p}(0)$, $\mathbf{p}(1)$, $\mathbf{p}^u(0)$ and $\mathbf{p}^u(1)$ are called are called *geometric coefficients*, and the terms F_1, F_2, F_3, and F_4 are called *blending functions*. Each blending function controls one of the four

boundary conditions. For example, $F_1(u)$ controls the endpoint at u = 0, but does not affect the endpoint at u = 1 or the tangent vectors at u = 0 or u = 1. Similarly, the blending function $F_4(u)$ determines the tangent vector at u = 1, but does not affect the other boundary conditions. Between these boundaries ($0 < u < 1$), the four functions F_1, F_2, F_3, and F_4 "blend" together, hence the name "blending function".

Matrix Representation

As was previously mentioned, parametric cubic curves are easily written in matrix form. The algebraic form in equation 96 can be rewritten as

$$\mathbf{p} = [u^3 \;\; u^2 \;\; u \;\; 1][a_3 \;\; a_2 \;\; a_1 \;\; a_0]^T \tag{111}$$

if $\mathbf{U} = [u^3 \;\; u^2 \;\; u \;\; 1]$ and $\mathbf{A} = [\mathbf{a}_3 \;\; \mathbf{a}_2 \;\; \mathbf{a}_1 \;\; \mathbf{a}_0]$ then equation (111) becomes

$$\mathbf{p} = \mathbf{U}\ \mathbf{A} \tag{112}$$

Similarly, the geometric form in equation 110 may be written as

$$\mathbf{p} = \mathbf{F}\ \mathbf{B} \tag{113}$$

where $\mathbf{F} = [F_1 \;\; F_2 \;\; F_3 \;\; F_4]$ and $\mathbf{B} = [\mathbf{p}_0 \;\; \mathbf{p}_1 \;\; \mathbf{p}_0' \;\; \mathbf{p}_1']^T$

Thus, matrix **A** represents the algebraic coefficients while matrix **B** contains the geometric coefficients. A relationship between the two matrices can be derived by further simplification of equation 113. The matrix **F** can be factored as

$$\mathbf{B} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{114}$$

The 4x4 matrix in equation 114 is called the *universal transformation matrix* denoted **M** Thus,

$$\mathbf{F} = \mathbf{U}\ \mathbf{M} \tag{115}$$

Substitution of equation 115 into equation 113 yields

$$\mathbf{p} = \mathbf{U}\ \mathbf{M}\ \mathbf{B} \tag{116}$$

Comparision of equation 116 with equation 112 shows that the algebraic and geometric coefficients are related by

$$\mathbf{A} = \mathbf{M}\ \mathbf{B} \tag{117}$$

or

$$\mathbf{B} = \mathbf{M}^{-1}\ \mathbf{A} \tag{118}$$

where

$$\mathbf{M}^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \tag{119}$$

Thus, the universal transform matrix **M** and its inverse $\mathbf{M}^{-1}$ allow easy conversion between the algebraic and geometric forms of a parametric cubic curve. In the remainder of this development, the most common form will be the geometric form

$$\mathbf{p}(u) = \mathbf{U}\ \mathbf{M}\ \mathbf{B} \tag{120}$$

Types of Parametric Cubic Curves

The geometric form of parametric cubic curves is directly applicable to the description of turbomachine blades where much of the geometry is known in terms of endpoints and their tangent vectors. The particular types of parametric cubic curves (in their geometric form) needed to model curves in a turbomachine blade are described below.

Four-Point Form

In some cases, a parametric curve must be specified to pass through four points instead of being specified in terms of the endpoints and their tangent vectors. For example, in a blade cross section, it might be desirable to fit a curve through the points defining the thickness distribution of the upper and lower surfaces of the cross section.

Suppose that it is desired to fit a parametric cubic curve through four points in space $[\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4]$. First, each point is assigned a value of the parametric variable u so that $(u_1 < u_2 < u_3 < u_4)$. In the following development,

$$u_1 = 0$$

$$u_2 = 1/3$$

$$u_3 = 2/3$$

$$u_4 = 1$$

then the problem is reduced to finding a matrix **K** which premultiplies the matrix of coordinates of the four points, producing the familiar geometric coefficient matrix **B**. That is,

$$[q_o \quad q_1 \quad q_0^u \; q_1^u] = \mathbf{K}[p_1 \quad p_2 \; p_3 \; p_4]^T \tag{121}$$

Mortenson [20] presents a derivation of the **K** -matrix and its inverse. The result is that **K** is a matrix of constant coefficients dependent only on the value of u assigned to each of the four points.

When the interval $u \in [0,1]$ is divided equally ($u_1 = 0, u_2 = 1/3, u_3 = 2/3, u_4 = 1$) then the K-matrix and its inverse $\mathbf{K}^{-1}$ are

$$\mathbf{K} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -11/2 & 9 & -9/2 & 1 \\ -1 & 9/2 & -9 & 11/2 \end{bmatrix} \tag{122}$$

$$\mathbf{K}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 20/27 & 7/27 & 4/27 & -2/27 \\ 7/27 & 20/27 & 2/27 & -4/27 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{123}$$

Conversion between the four-point and geometric forms is accomplished by

$$\mathbf{B} = \mathbf{K}[p_1 \; p_2 \; p_3 \; p_4] \tag{124}$$

and

$$[p_1 \; p_2 \; p_3 \; p_4] = \mathbf{K}^{-1}\mathbf{B} \tag{125}$$

Straight Lines

Straight lines are used to define the leading and trailing edges on a turbomachine blade, and even though they can be easily defined nonparametrically, it is more efficient to define all geometry (including lines) parametrically.

The geometric form of a parametric cubic curve fitting a straight line between points $\mathbf{p}_0$ and $\mathbf{p}_1$ is

$$\mathbf{p} = \mathbf{B} \quad [\mathbf{p}_0 \quad \mathbf{p}_1 \quad (\mathbf{p}_1 - \mathbf{p}_0) \quad (\mathbf{p}_1 - \mathbf{p}_0)]^T \tag{126}$$

where **F** is the matrix of blending functions previously defined.

Circular Arcs

The final part of the blade geometry to be modeled with a parametric cubic curve is the camberline. The most common camberline shapes--circular and parabolic arcs--both belong to the conic family and are discussed at length in Mortenson [20]. The results for the circular arc are presented below.

A circular arc may be specified in many ways. In this development, the camberline is specified by the two endpoints $\mathbf{p}_0$ and $\mathbf{p}_1$ and the center of curvature $\mathbf{p}_c$. Following the standard geometric form for a parametric cubic curve

$$\mathbf{p} = \mathbf{F} \quad [\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_0^u \quad \mathbf{p}_1^u]^T \tag{127}$$

$\mathbf{p}_0$ and $\mathbf{p}_1$ are known, but their tangent vectors $\mathbf{p}_0^u$ and $\mathbf{p}_1^u$ must be expressed in terms of the known quantities. Mortenson [20] shows that these tangent vectors are

$$\mathbf{p}_0^u = \frac{2}{\cos\theta(1 + \cos\theta)}(\mathbf{p}_1 - \mathbf{p}_0 + 2\sin^2\theta(\mathbf{p}_0 - \mathbf{p}_c)) \tag{128}$$

$$p_1^u = \frac{2}{\cos\theta(1 + \cos\theta)}(p_1 - p_0 + 2\sin^2\theta(p_c - p_1)) \tag{129}$$

where 0 is the included angle $(\beta'_2 - \beta'_1)$ and **F** is the matrix of blending functions previously discussed.

Parametric Bicubic Patches

The simplest way to model a surface is with a parametric patch, which may be regarded as a limited region of a larger surface. Mortenson defines a patch as a "curve-bounded collection of points whose coordinates are given by continuous, two-parameter, single-valued mathematical functions." The form of a parametric patch is

$$\begin{aligned} x &= x(u,w) \\ y &= y(u,w) \\ z &= z(u,w) \end{aligned} \tag{130}$$

where $u,w \in [0,1]$

A major difference between a parametric curve and a parametric patch is the need for two parametric variables (u,w) to define a patch, instead of just one. The patch is created by holding one variable constant while the other varies between 0 and 1. By repeating this process, a net of curves is created. If all of the curves are cubic, then the patch is called *bicubic.*

Parametric bicubic patches may be expressed in algebraic or geometric forms, just like parametric cubic curves. However, more boundary conditions are required to define a patch, resulting in 48 geometric coefficients. Referring to Figure 15 the boundary conditions are

- four corner points:

  $p(0,0)$, $p(0,1)$, $p(1,0)$, and $p(1,1)$

- four parametric cubic curves at the boundaries:

  $p(0,w)$, $p(1,w)$, $p(u,0)$, and $p(u,1)$

- four twist vectors or cross derivatives at the boundaries:

  $p^{uw}(0,0)$, $p^{uw}(0,1)$, $p^{uw}(1,0)$, and $p^{uw}(1,1)$

The twist vectors influence the interior shape of the patch. The simplest patch is a *Ferguson* patch in which all the twist vectors are zero. The boundary conditions are set up in a 4x4 matrix **B** as follows

$$\mathbf{B} = \begin{bmatrix} p_{00} & p_{01} & p_{00}^{w} & p_{01}^{uw} \\ p_{10} & p_{11} & p_{10}^{w} & p_{11}^{w} \\ p_{00}^{u} & p01^{u} & p_{00}^{uw} & p_{01}^{uw} \\ p_{10}^{u} & p_{11}^{u} & p_{10}^{uw} & p_{11}^{uw} \end{bmatrix} \tag{131}$$

so that the geometric form of a bicubic parametric patch is similar to a parametric cubic:

$$p(u,w) = \begin{bmatrix} F_1(u) & F_2(u) & F_3(u) & F_4(u) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{00}^{w} & p_{01}^{uw} \\ p_{10} & p_{11} & p_{10}^{w} & p_{11}^{w} \\ p_{00}^{u} & p_{01}^{u} & p_{00}^{uw} & p_{01}^{uw} \\ p_{10}^{u} & p_{11}^{u} & p_{10}^{uw} & p_{11}^{uw} \end{bmatrix} \begin{bmatrix} F_1(w) \\ F_2(w) \\ F_3(w) \\ F_4(w) \end{bmatrix} \tag{132}$$

where **F** (u) and **F** (w) are the same blending functions discussed in relation to the parametric cubic curves.

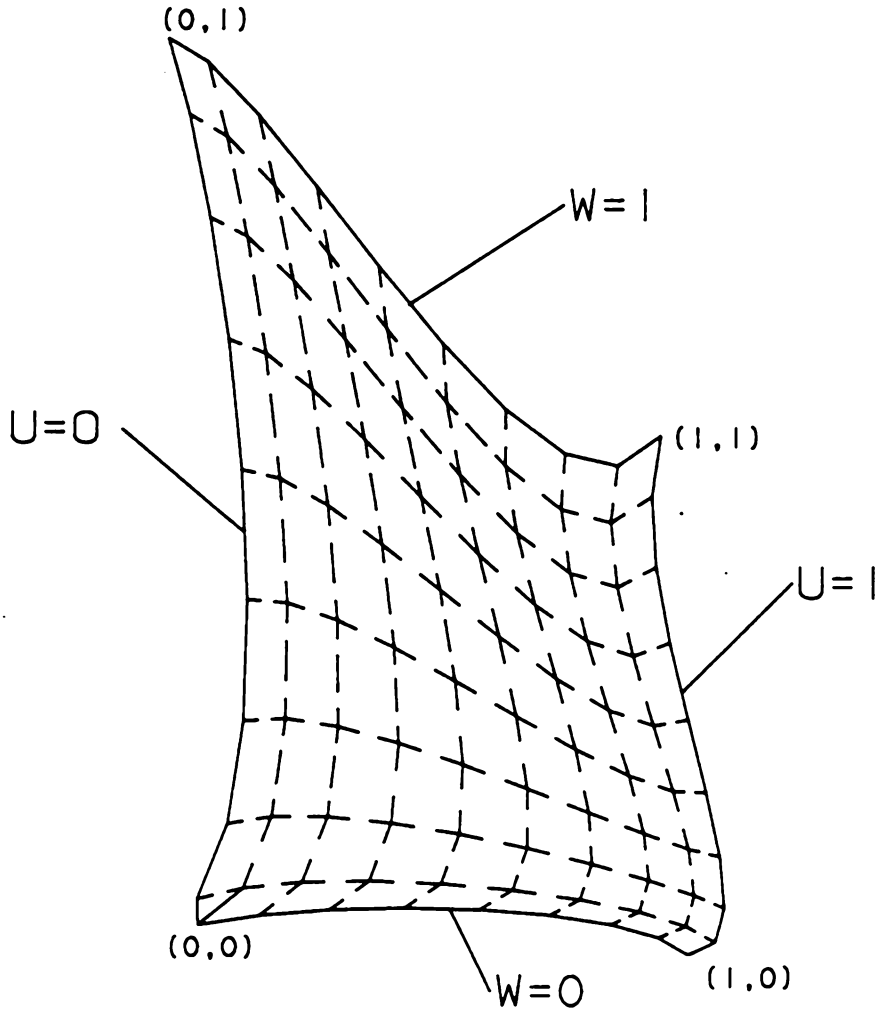

Figure 15. Nomenclature for Bicubic Patches.

In matrix form, equation 132 is

$$p(u,w) = \mathbf{F}(u)\ \mathbf{B}\ \mathbf{F}(w)^T \tag{133}$$

or

$$\mathbf{p} = \mathbf{U}\ \mathbf{M}\ \mathbf{B}\ \mathbf{M}^T\ \mathbf{W}^T \tag{134}$$

In the case of fan blades, the surface to be modeled lies between adjacent cross sections on either the upper or lower blade surface, as shown in Figure 16. Two endpoints lie on the leading edge and two endpoints lie on the trailing edge. The four boundary curves are all parametric cubic curves previously described. The u = 0 and u = 1 boundary curves are parametric cubic curves fitted between the points describing the thickness distribution of the cross sections. The w = 0 and w = 1 boundary curves are straight lines defining the the leading and trailing edges. The blade surface is thus represented by successive iterations of holding one variable constant (at some value between zero and one) and allowing the other variable to vary. The result is a mesh of parametric cubic curves defining the blade cross sections. The entire blade is then modeled by repeating this process, as shown in Figure 17.

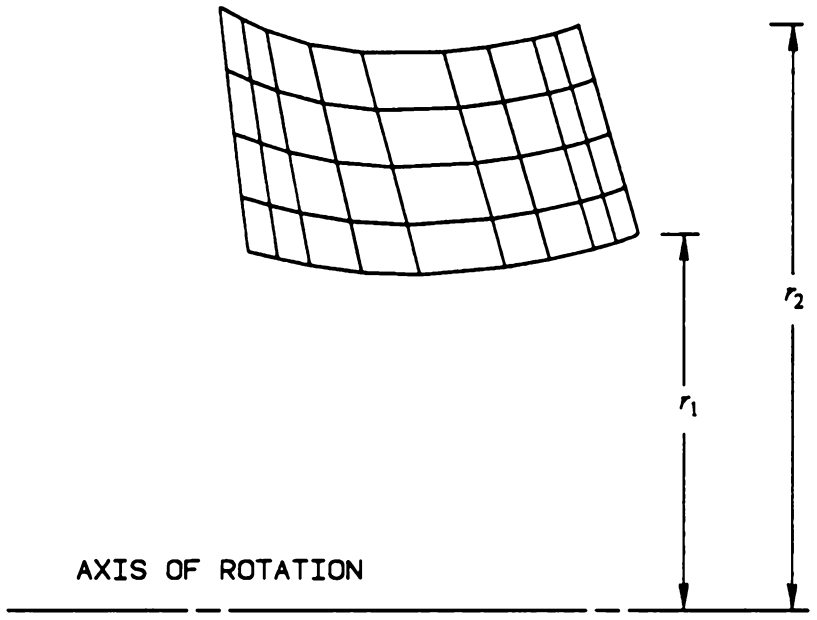

Figure 16. Parametric Bicubic Patch on an Axial-Flow Fan Blade

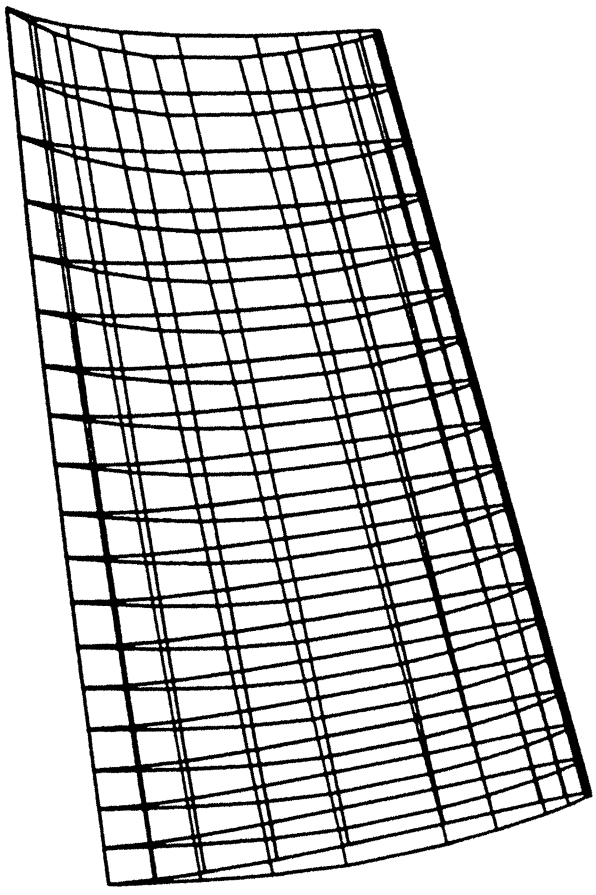

Figure 17. Geometric Model of an Axial-Flow Fan Blade

Graphics Support

CADAM

The graphical support of the work presented in this thesis is provided by CADAM, or Computer-Graphics Augmented Design and Manufacturing System. Originally developed by Lockheed Corporation in the 1960's, CADAM has become one of the most popular computer-aided design, computer-aided manufacturing (CAD/CAM) software packages. CADAM consists of a set of computer programs with a common database. Its primary purpose is the production of engineering models on a computer terminal, serving as a useful design, drafting, and manufacturing tool.

Geometry Interface

A particularly useful feature of CADAM is the *Geometry Interface Module* which allows a user to directly access and modify the CADAM database in a batch or interactive environment using utility programs. Basically, this means that a user can create, modify, and retrieve models stored in the CADAM database--by executing a user-written applications program instead of manually initiating the graphics.

The CADAM geometry interface is made up of four functions.

CADET retrieves models from a CADAM database.

CADCD generates elements which are added directly to the CADAM database as a CADAM model.

CADMACGM allows the user to write his own interactive program using CADAM functions.

SCLIB produces a list of models in a user's drawing file.

The function CADCD provides the primary graphic support in this thesis. It consists of a set of subroutines invoked by Fortran CALL statements producing CADAM elements in a CADAM model. CADCD enables the automatic generation of new drawings, the modification of existing drawings, and the addition of details. All of this is accomplished by writing an applications program containing the appropriate CADCD CALL statements and linking the applications program into the CADCD module, as the main routine.

Appendix B contains descriptions of the CADCD routines used by the FAN3D program. The basic structure is:

1. Routine CADST initializes a CADAM model.

2. Routine BEGVU initializes a CADAM model view.

3. Various routines draw geometric primitives like points, lines, circles, and splines in the current model view. The primitives used in FAN3D are described in Appendix B.

4. Routine CADFIL files the CADAM model.

Program Description

The application program FAN3D follows the same path as the progression of this thesis. FAN3D is written in Fortran77 and its graphics support is provided by the CADCD component of the Geometry Interface in in CADAM, as previously discussed. Figure 18 shows a flowchart of the overall structure of FAN3D.

The aerodynamic parameters to be initially specified are

- mass flow rate
- inlet pressure
- inlet temperature
- total pressure rise for stage
- rotational speed
- density of blade material

The geometric parameters to be initially specified are

- hub radius
- tip radius
- blade solidity
- ratio of maximum camber to chord

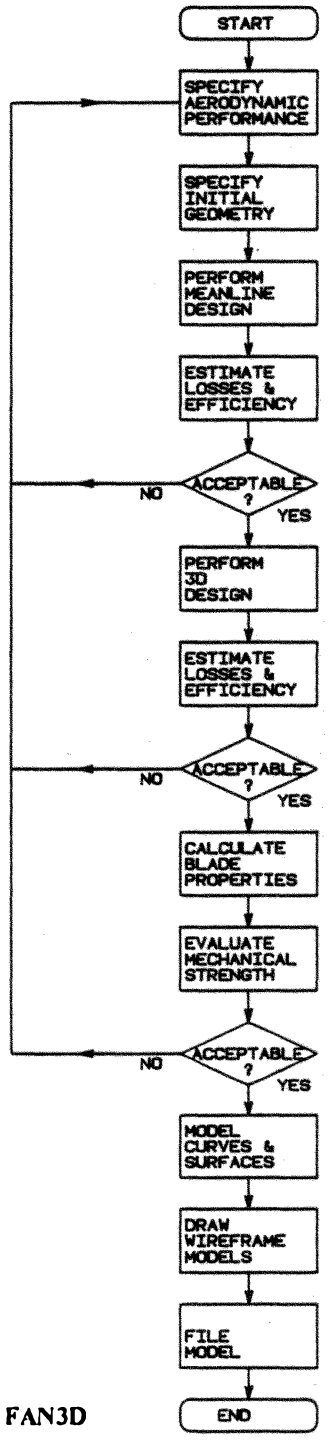

Figure 18. Flowchart for Program FAN3D

- ratio of blade span to chord
- ratio of blade clearance to blade height

The program FAN3D was written in modular form so that modifications or addditions are easy. The subroutines in FAN3D are

ARCPC computes the geometric properties of a parametric cubic equation of a circular arc.

BF computes the geometric blending functions for a specified value of the parametric variable.

BFU computes the first derivative of the geometric blending functions for a specified value of the parametric variable.

BFUU computes the second derivative of the geometric blending functions for a specified value of the parametric variable.

BLPROP computes the cross sectional properties at specified points along the blade length.

COORD computes the coordinates of points on the upper and lower surfaces of a NACA 65-series airfoil.

CSEC draws parametric cubic curves for the blade cross sections.

CURVE converts a 4-point form parametric cubic equation to the geometric form.

FPTOG computes geometric coefficients of a parametric cubic equation, given four points on the curve.

FVORTX	computes flow and blade angles of an axial-flow fan blade based on free vortex flow.
GEOBLD	computes the geometry required to model the blade cross sections.
LOSS3D	estimates the losses in a three-dimensional axial-flow blade.
MEAN	calculates the blade and flow angles at the mean radius for an axial-flow fan blade.
MLOSS	estimates blade performance based on meanline geometry only.
NONDIM	computes nondimensional parameters describing aerodynamic operation.
OUT1	outputs results of calculations.
OUT2	outputs results of calculations.
OUT3	outputs results of calculations.
OUT4	outputs results of calculations.
PCRV	computes the coordinates of a point on a parametric cubic curve, given the geometric form and a specified value of the parametric variable.
PSRF	computes the coordinates of a point on a bicubic parametric patch using the geometric form.
PPSRF	computes the coordinates of a rectangular array of points on a bicubic patch at specified increments of the two parametric variables.

REPARM	reparameterizes a parametric cubic curve.
ROTLN	rotates and draws a line.
ROTMAT	calculates rotation matrix.
ROTPT	rotates and draws a point.
ROTSRF	rotates and draws a blade surface.
ROTVAR	calculates parameters needed to rotate an element.
SURF	draws blade surfaces.
VMAG	computes the magnitude of vectors.

Examples

As an illustration of how the program FAN3D operates, consider the need to design a fan meeting the following aerodynamic specifications:

- mass flow rate = 75 kg/s
- inlet pressure = 100 kPa
- inlet temperature = 300 K
- total pressure rise for stage = 2.5 kPa
- rotational speed = 180 rad/s
- density of blade material = 500 kg/m^3

and the following geometric specifications:

- hub radius = 0.25 m
- tip radius = 0.50 m
- blade solidity = 1.5
- ratio of maximum camber to chord = 0.5
- ratio of blade span to chord = 1.0
- ratio of blade clearance to blade height = 0.002

Using these specifications, FAN3D would produce results of its calculations, as shown in Appendix D. In addition, FAN3D would produce two models in the CADAM database. The first is a detailed drawing of a single blade, represented by bicubic parametric surfaces. Figures 19 through 22 show the blade, rotated through 45-degree increments about the axis of the blade. The second model produced in CADAM is the complete blade set, rotated about the axis of rotation. These blades are much simpler in representation than the single blade because of model-size contraints. Figures 23 through 27 show the blade set rotated around line AB. Figures 28 and 29 show the blade set in Figure 27 rotated around line CD.

Using the information presented in the preceding figures, with that presented in Appendix D, the designer has a number of choices. If all of the results are satisfactory, then he might proceed with this design to more detailed stress calculations, vibration analysis, or manufacture. If the results are not completely satisfactory, the designer might:

- revise aerodynamic performance specifications
- revise initial geometric specifications
- choose a new method for estimating efficiency
- choose a new tangential velocity distribution (perhaps forced vortex instead of free vortex).

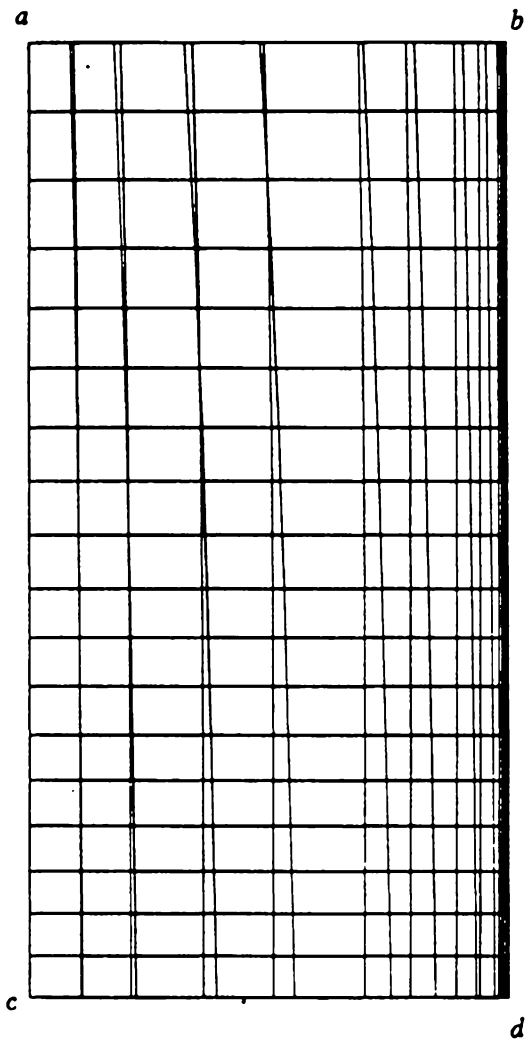

Figure 19. Single Blade, Initial Position

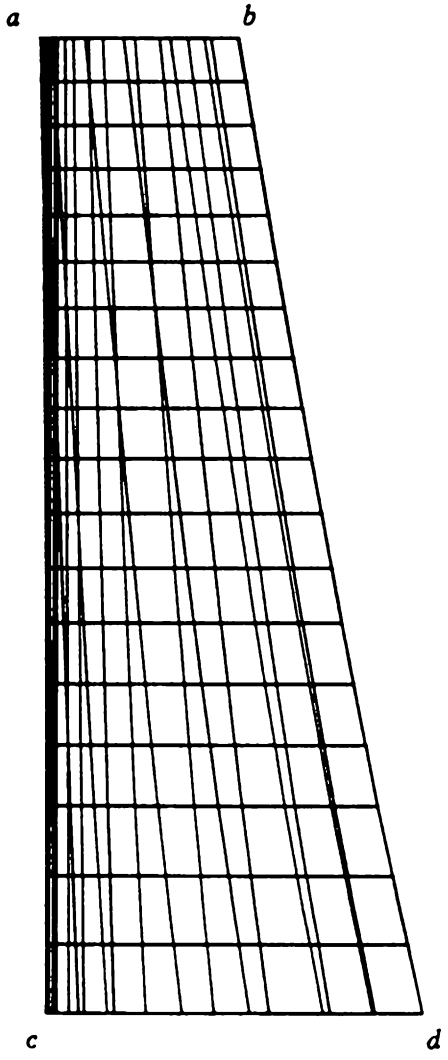

Figure 20. Single Blade, Rotated 45 Degrees

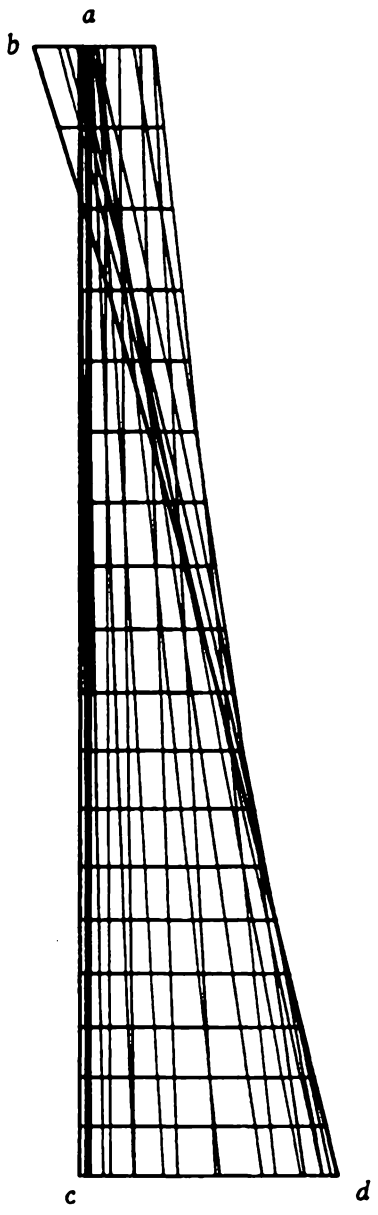

Figure 21. Single Blade, Rotated 90 Degrees

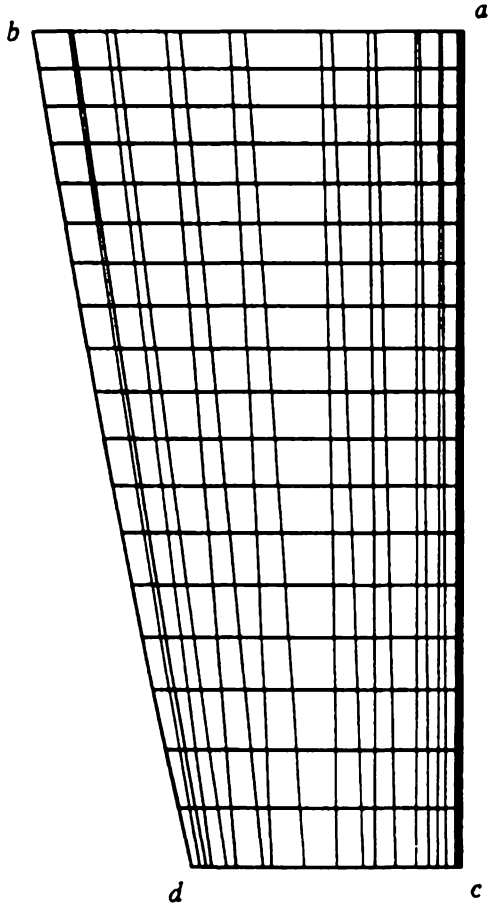

Figure 22. Single Blade, Rotated 135 Degrees

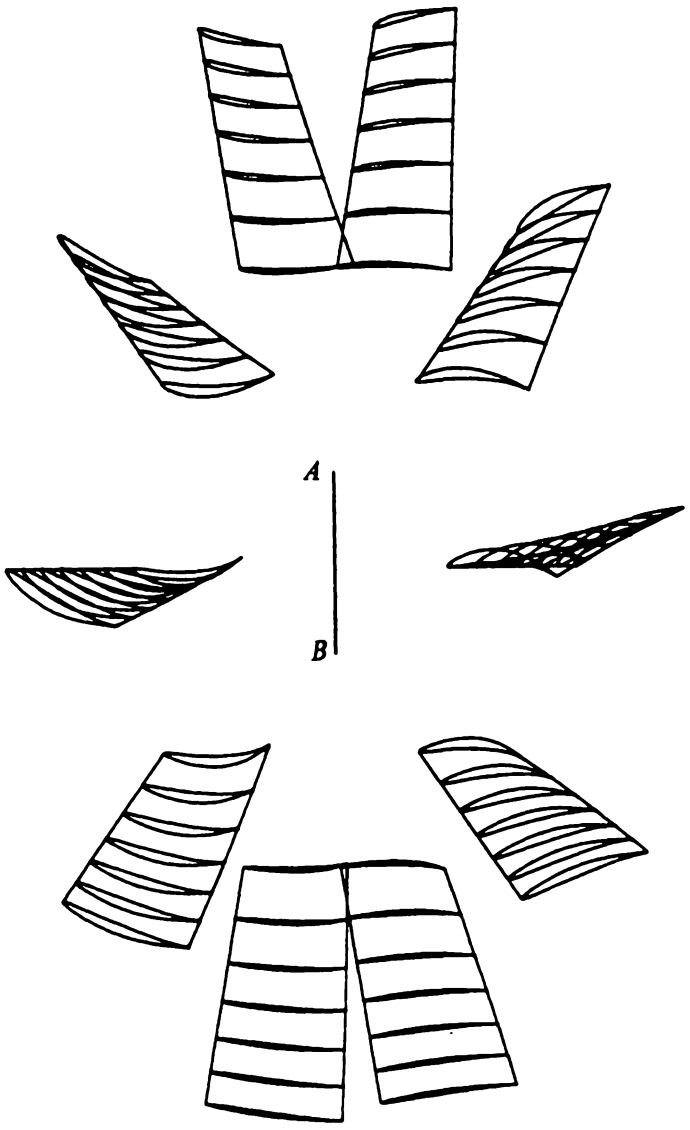

Figure 23. Blade Set, Rotated 30 Degrees About Line AB

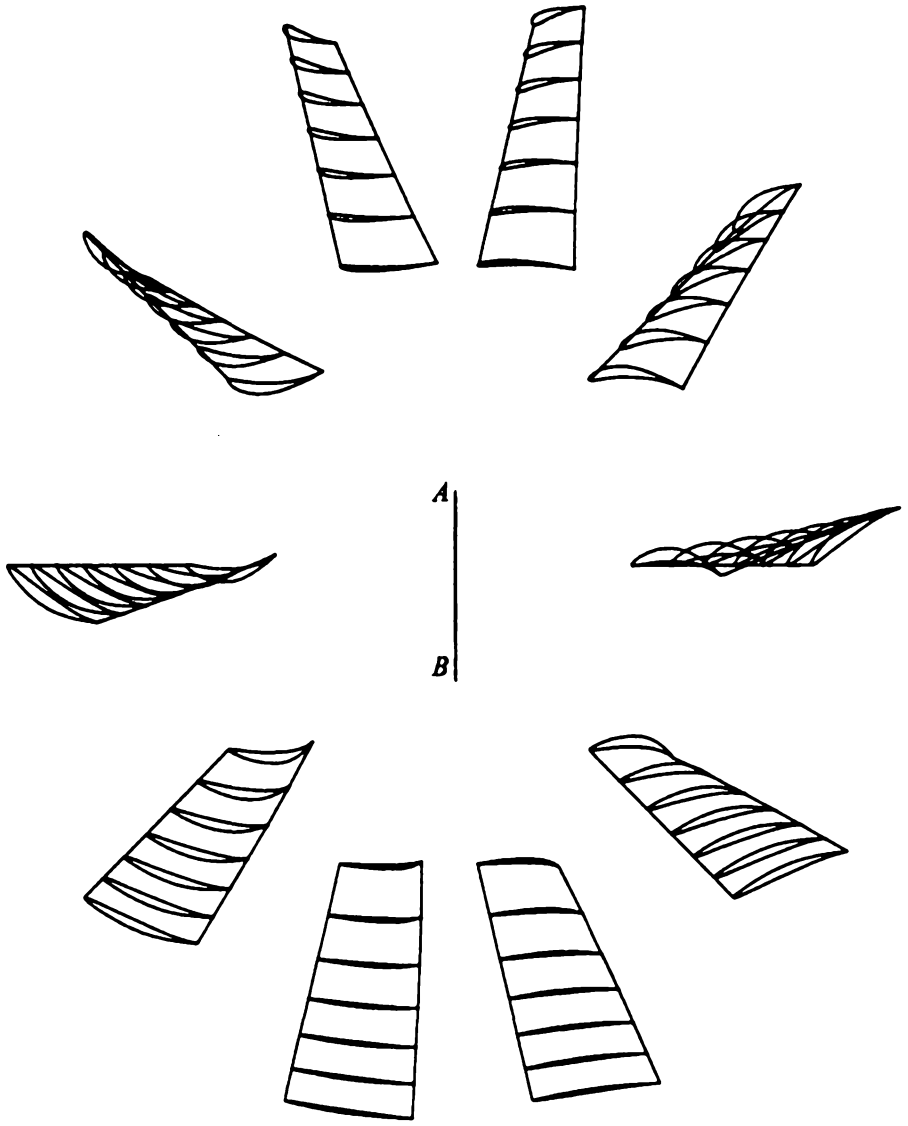

Figure 24. Blade Set, Rotated 45 Degrees About Line AB

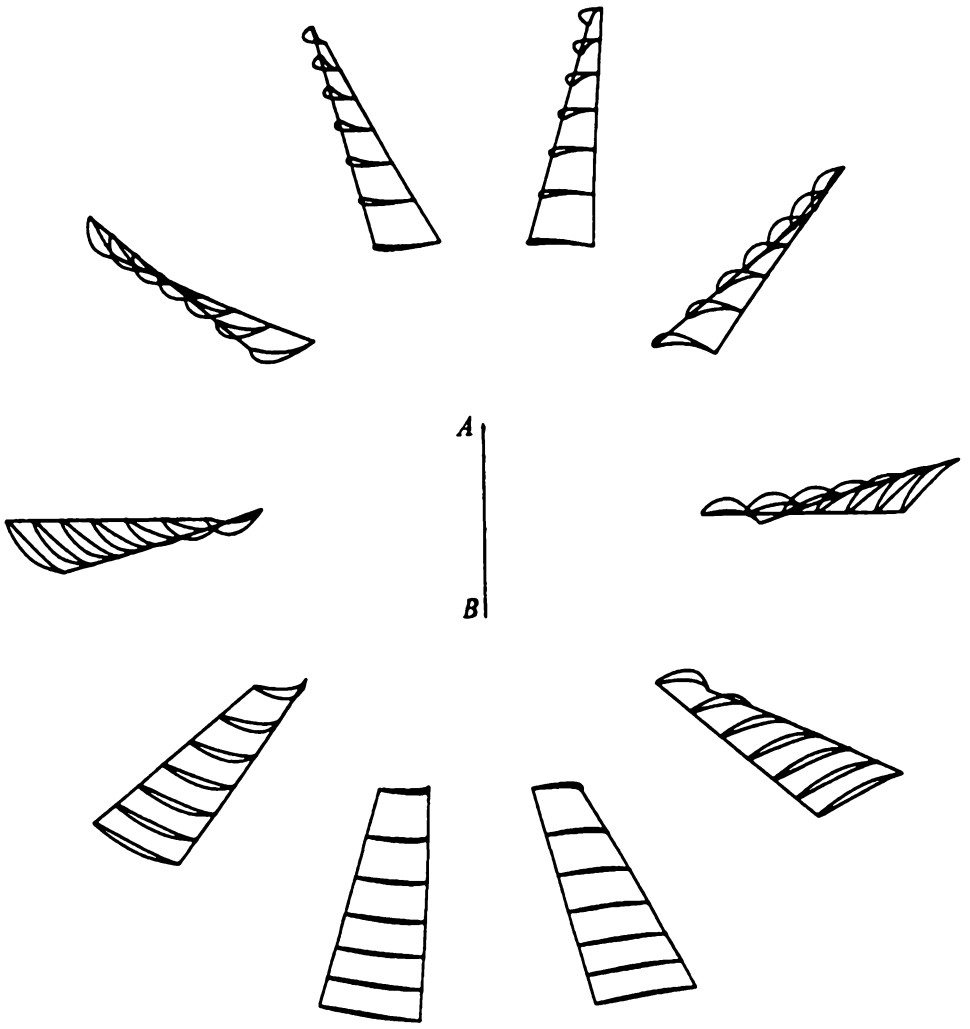

Figure 25. Blade Set, Rotated 60 Degrees About Line AB

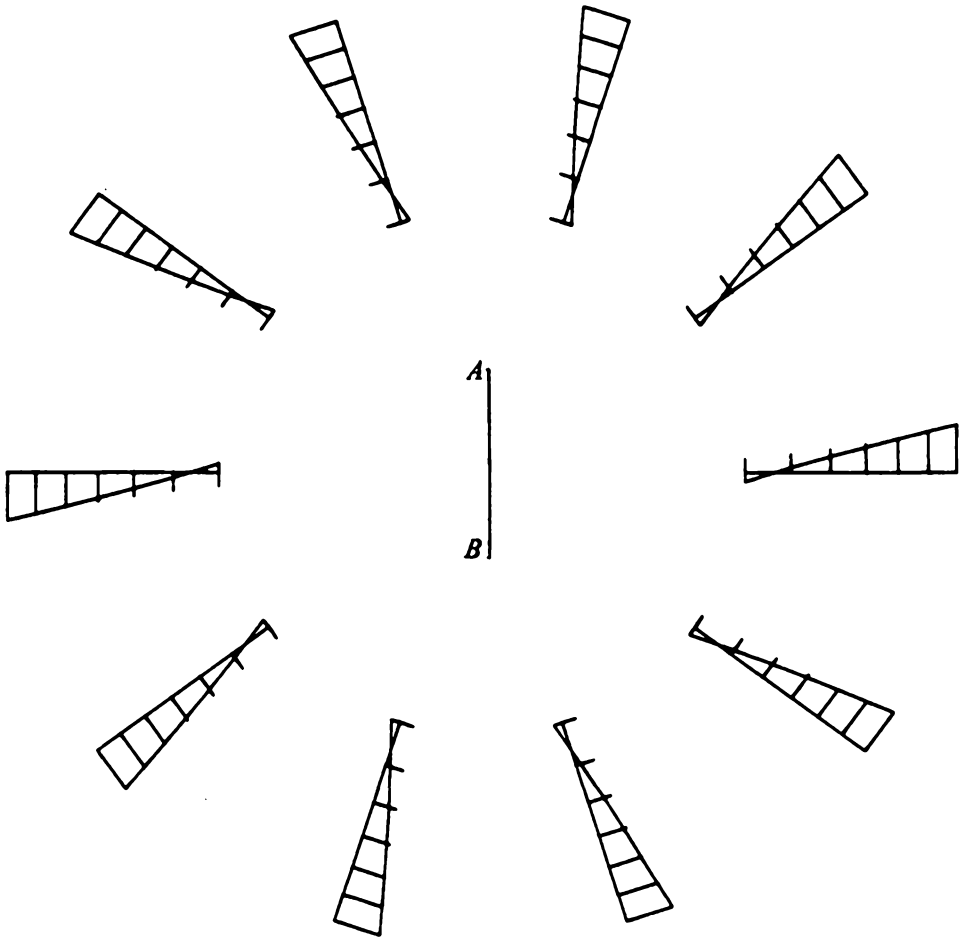

Figure 26. Blade Set, Rotated 90 Degrees About Line AB

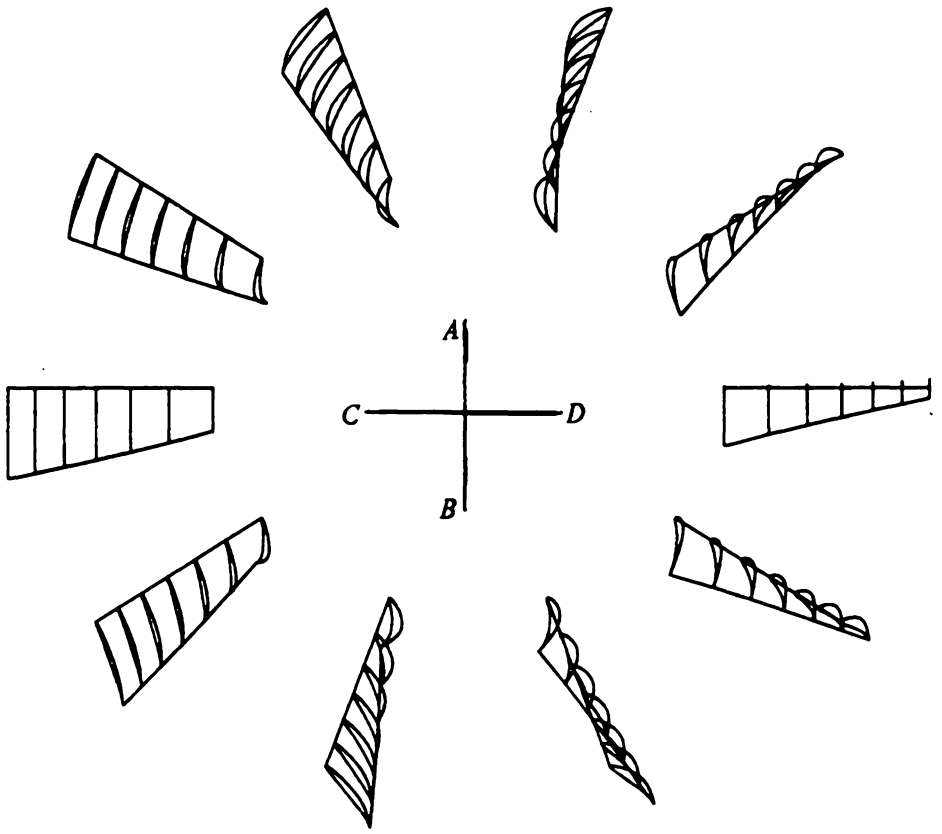

Figure 27. Blade Set of Figure 27, Rotated 30 Degrees About Line CD

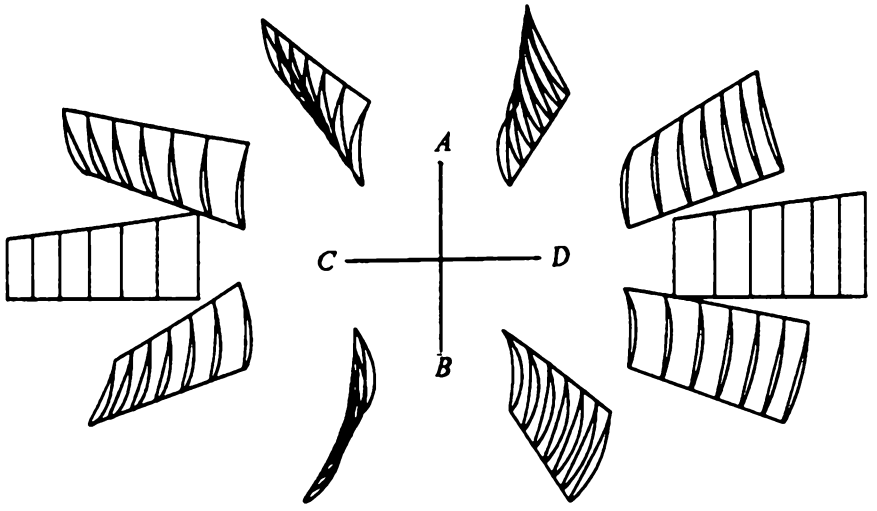

Figure 28. Blade Set of Figure 27, Rotated 60 Degrees About Line CD

Conclusions

This thesis has examined the application of computer-aided design techniques to the field of turbomachinery. In particular, geometric modeling and automatic model generation via the CADAM Geometry Interface have been used to aid the designer of axial-flow fans.

The most important conclusion to be drawn from this work is that computer-aided design techniques are useful throughout the process of designing an axial-flow fan. From a computational point of view, computers are essential in order to solve sets of equations requiring iteration. Additionally, computers are well-suited to the iteration needed to optimize design parameters. Beyond these more traditional applications of computer-aided design techniques to turbomachinery, this thesis has demonstrated how geometric modeling techniques can be used to completely describe a blade's geometry by fitting parametric bicubic curves and surfaces to a discrete number of blade cross sections. And finally, the automatic generation of models by the CADCD component of the Geometry Interface in CADAM has been demonstrated.

The CADCD portion of this geometry interface has several advantages. First, it is operated by Fortran CALL statements to the CADCD subroutines. With an understanding of the purposes and constraints of these subroutines, geometry can be easily represented, without learning a new computer language. Also, since the CADCD CALL statements can be included directly in the application Fortran program, it is possible to document graphically the iterations made to optimize a design. When the final design has been chosen, the geometry is located in a central database, where programs analyzing structural integrity, vibrational response, or manufacturing could access the blade model, and finally, numerical control instructions can be generated using CADAM to cut these blades.

However, with all of these advantages of computer-aided design, it will not completely replace the knowledge and intuition of an experienced turbomachine designer. For example, while three-dimensional graphics often provide a new way to represent data, it can often be misleading. Depending on its orientation relative to the viewer, a blade sometimes might appear straight, moderately twisted, or severely twisted. Experience with the computer graphics is the only solution to this problem.

Future work on the application of computer-aided design techniques to turbomachinery might include:

- expert systems
- interactive graphics
- solid modeling.

Appendix A.

Coordinates for NACA 65-010 Basic Thickness Form

(Stations and Ordinates in Percent of Chord)

Station (x/c)	Ordinate (y/c)
0.	0.
0.5	0.752
0.75	0.890
1.25	1.124
2.5	1.571
5.0	2.222
7.5	2.709
10.0	3.111
15.0	3.746
20.0	4.218
25.0	4.570
30.0	4.824
35.0	4.982
40.0	5.057
45.0	5.029
50.0	4.870
55.0	4.570
60.0	4.151
65.0	3.627
70.0	3.038

75.0	2.451
80.0	1.847
85.0	1.251
90.0	0.749
95.0	0.354
100.0	0.150

NOTE:

See nomenclature defined in Figure 11

Appendix B.

CADCD Subroutines

CADST

Initialize a model

CALL CADST (DRAWID, EMPNO, GROUP)

DRAWID A five-word array containing the drawing file.

EMPNO A two-word array containing the employee ID.

GROUP A one-word array containing the group ID.

BEGVU

Begin a trap for view geometry

CALL BEGVU (IOPT, IDVU, XYZ, XVECTR, YVECTR, ZVECTR, *E)

IOPT An option flag which must be set to one.

IDVU Four characters, of which the last two are used, 'xxid'

XYZ An array XYZ(3) representing the X,Y,Z location of the view

XVECTR An array XVECTR(3) representing the A,B,C direction cosines of the new X-axis

YVECTR An array YVECTR(3) representing the A,B,C direction cosines of the new Y-axis

ZVECTR An array YVECTR(3) representing the A,B,C direction cosines of the new Z-axis

***E** - Error--A view is already started, or no more space for views

PT3D

Create a 3-D point

CALL PT3D (XYZ, *E)

XYZ An array XYZ(3) containing the X,Y,Z absolute coordinates of the point.

***E** Error not as yet specified.

LINE3D

Create a 3D line

CALL LINE3D (XYZ1, XYZ2, &E)

XYZ1 An array XYZ(3) containing the X,Y,Z absolute coordinates of the first endpoint of the line

XYZ2 An array XYZ(3) containing the X,Y,Z absolute coordinates of the second endpoint of the line

***E** Error, length of line less than 0.001

SPLINE

Create a 3-D spline

CALL SPLINE (N,M,NDmXYZ,ABC,S,A,B,R,*E)

N An integer value representing the number of input points

M An integer value specifying the number of dimensions in space.

M = 1 denotes a singly dimensioned spline. Flange angle information (in radian) is input through the XYZ array (see below) rather than X-coordinate data.

M = 2 denotes a two-dimensional spline.(X,Y) coordinates are input through the XYZ array.

M = 3 denotes a three-dimensional spline. (X,Y) coordinates are input through the XYZ array.

ND An integer value specifying end conditions

ND = 0 for curved ends (slopes unknown)

ND = 1 for first slope supplied in ABC(1,1) and ABC(2,1)

ND = 2 for last slope supplied in ABC(1,N) and ABC(2,N)

ND = 3 for first and last slopes supplied

XYZ(M,N) A floating point array containing spline point coordinate data, if M > 1. If M = 1, the XYZ array holds flange angle information.

S(N) A floating point array of chord lengths for the bays defined by the input data.

ABC(M,N) A floating point array of spline tangent vector components at the input points. If M = 1, this array contains flange angle derivative approximations.

A A temporary floating point working area of length N.

B A temporary floating point working area of length N.

R A temporary floating point working area of length N.

***E** Denotes a label in the calling program for error return.

NAME

Provide a name for the last primitive or group of primitives created

CALL NAME (NUMBER,*E)

NUMBER A unique integer returned by the routine which identifies the last element created.

***E** Denotes a label in the calling program for error return.

CADFIL

File a drawing

CALL CADFIL (IOPT,NOGOOD,IDUMMY)

IOPT = 0, file drawing

= 1, retrieve drawing

= 2, overfile drawing

NOGOOD An integer to be tested after filing. If *NOGOOD* ≠ 1 then the drawing is not filed.

IDUMMY A dummy variable not used in current release.

Appendix C.

FAN3D: Program Listing

```
C ************************************************************
C
C
C          FREE VORTEX BLADE MODELING
C
C
C     This program models three-dimensional axial-flow fan blades
C
C     Created by L.M. Powers
C     3 August 1986
C
C ************************************************************
C
C
C
C     DEFINE VARIABLES
C
C    AC       DISTANCE TO PT. OF MAX CAMBER
C    AM1      MEAN INLET ABSOLUTE FLOW ANGLE (r)
C    A1(I)    INLET ABSOLUTE FLOW ANGLE AS FN OF RADIUS (r)
C    AM2      MEAN OUTLET ABSOLUTE FLOW ANGLE (r)
C    A2M(I)   OUTLET ABSOLUTE FLOW ANGLE AS FN OF RADIUS (r)
C    B        BLOCKAGE FACTOR
```

```
C   BR1M     MEAN INLET BLADE ANGLE (r)
C   BR1(I)   INLET BLADE ANGLE AS FN OF RADIUS (r)
C   BR1MD    INLET BLADE ANGLE AS FN OF RADIUS (DEG)
C   BR2M     MEAN OUTLET BLADE ANGLE (r)
C   BR2(I)   OUTLET BLADE ANGLE AS FN OF RADIUS (r)
C   BR2MD    OUTLET BLADE ANGLE AS FN OF RADIUS (DEG)
C   B1M      MEAN INLET RELATIVE FLOW ANGLE (r)
C   B2M      MEAN OUTLET RELATIVE FLOW ANGLE (r)
C   DEL      DEVIATION ANGLE (r)
C   DPT      CHANGE IN TOTAL PRESSURE (kPa)
C   DR       DIFFUSION RATIO
C   EF1      GUESSED EFFICIENCY
C   EF2      CALCULATED EFFICIENCY
C   GMA      WORK DONE FACTOR
C   HC       SPAN/CHORD RATIO
C   INMED    INTERMEDIATE VARIABLE USED IN LOSS CALCULATION
C   K        COUNTER
C   MF       MASS FLOWRATE (kg/s)
C   N        COUNTER
C   P1       INLET PRESSURE (kPa)
C   PT1      INLET TOTAL PRESSURE (kPa)
C   PT2      OUTLET TOTAL PRESSURE (kPa)
C   RGAS     GAS CONSTANT (kJ/kg K)
C   RHO      DENSITY--ASSUMED CONSTANT (kg/m**3)
C   RM       MEAN RADIUS (m)
C   RI       INNER INLET RADIUS (m)
C   RO       OUTER INLET RADIUS (m)
C   TH2C     WAKE MOMENTUM THICKNESS RATIO
```

```
C    T1       INLET TEMPERATURE (K)
C    UM        ANGULAR VELOCITY AT MEAN RADIUS (m/s)
C    U(I)     ANGULAR VELOCITY AS FN OF RADIUS (m/s)
C    VX1       INLET AXIAL VELOCITY (m/s)
C    VX2       OUTLET AXIAL VELOCITY (m/s)
C    V2       OUTLET VELOCITY (m/s)
C    W        ANGULAR VELOCITY (r/s)
C    WA        COEFFICIENT FOR LOSSES FROM ANNULUS WALLS
C    WP        COEFFICIENT FOR PROFILE LOSSES FOR BDY LAYER ON AIRFOIL
C    WS       COEFFICIENT FOR SECONDARY FLOW LOSSES
C    WROT      TOTAL COEFFICIENT FOR ROTOR LOSSES
C    WSTAT     TOTAL COEFFICIENT FOR STATOR LOSSES
C    W1       INLET RELATIVE VELOCITY (m/s)
C
C
C
*************************************************************************
*
*    Main
*
*    Purpose:     set up maximum dimensions of variable arrays and
*                 and call the controlling subroutine
*
*************************************************************************
     REAL BR1(25),BR2(25),R(25),A1(25),A2(25),U(25),BR1D(25),
    $B2(25),RARC(25),ST(25,3),ED(25,3),CT(25,3),BR2D(25)
    $,TOP(25,26,3),BOT(25,26,3),AREA(25),CG1(25,3),CG(25,3),F(25),
    $IXX(25),IYY(25),IXX0(25),IYY0(25),F0(25),IMAX(25),IMIN(25),
```

```
     $THETA(25),BS(25,26,3),TS(25,26,3),CSTRSS(25),
     $COARC(25,4,3),COTOP1(25,4,3),COTOP2(25,4,3),COTOP3(25,4,3),
     $COTOP4(25,4,3),COBOT1(25,4,3),COBOT2(25,4,3),COBOT3(25,4,3),
     $COBOT4(25,4,3),CAM(25,26,3),CIR(25,26,3)
      INTEGER IBR1D(25),IBR2D(25),IBR1M(25),IBR2M(25),IBR1S(25),
     $IBR2S(25)
      NSEC = 3
      NS = 2*NSEC + 1
      CALL CONTRL(NS,BR1,BR2,R,A1,A2,U,BR1D,BR2D,B2,
     $IBR1D,IBR2D,IBR1M,IBR2M,IBR1S,IBR2S,RARC,ST,ED,CT,
     $TOP,BOT,AREA,CG1,CG,F,IXX,IYY,IXX0,IYY0,F0,IMAX,IMIN,THETA
     $,TS,BS,COARC,COTOP1,COTOP2,COTOP3,COTOP4,COBOT1,COBOT2,COBOT3,
     $COBOT4,CAM,CIR,CSTRSS)
      STOP
      END
*************************************************************************
*************************************************************************
*************************************************************************


      SUBROUTINE CONTRL(NS,BR1,BR2,R,A1,A2,U,BR1D,BR2D,B2,
     $IBR1D,IBR2D,IBR1M,IBR2M,IBR1S,IBR2S,RARC,ST,ED,CT,
     $TOP,BOT,AREA,CG1,CG,F,IXX,IYY,IXX0,IYY0,F0,IMAX,IMIN,THETA
     $,TS,BS,COARC,COTOP1,COTOP2,COTOP3,COTOP4,COBOT1,COBOT2,COBOT3,
     $COBOT4,CAM,CIR,CSTRSS)
      CHARACTER DRAWID*20,USERID*8,GROUP*4,IDVU*4
      INTEGER IBR1D(NS),IBR1M(NS),IBR1S(NS)
      INTEGER IBR2D(NS),IBR2M(NS),IBR2S(NS)
```

```
      REAL BR1(NS),BR2(NS),R(NS),A1(NS),A2(NS),U(NS)
      REAL BR1D(NS),BR2D(NS),B2(NS)
      REAL RARC(NS),TANDIF(10)
      REAL ST(NS,3),ED(NS,3),CT(NS,3)
      REAL XVCTR(3),YVCTR(3),ZVCTR(3),XYZ(3)
      REAL AREA(NS),CG1(NS,3),CG(NS,3),F(NS),IXX(NS),IYY(NS),IXX0(NS)
      REAL IYY0(NS),F0(NS),IMAX(NS),IMIN(NS),THETA(NS),TS(NS,26,3)
      REAL BS(NS,26,3),CSTRSS(NS)
      REAL PT(3),PTA(3),PTB(3),CPT(3)
      REAL CAM(NS,26,3),TOP(NS,26,3),BOT(NS,26,3)
      REAL COARC(NS,4,3),OUT(4,3)
      REAL COTOP1(NS,4,3),COTOP2(NS,4,3),COTOP3(NS,4,3),COTOP4(NS,4,3)
      REAL COBOT1(NS,4,3),COBOT2(NS,4,3),COBOT3(NS,4,3),COBOT4(NS,4,3)
      REAL MF,INMED
      DATA IDVU/'IDPV'/,XVCTR/1.,0.,0./,YVCTR/0.,1.,0./,ZVCTR/0.,0.,1
     $./,XYZ/0.,0.,0./,NOUT/6/
      DATA DRAWID/'BLADE,ROTATE'/,USERID/'TURBO'/,GROUP/'ME'/
**
********************** define constants *********************************
**
      PI = 3.14159
      RGAS = 0.287
      B = 0.95
**
********************** define input *************************************
**
       MF = 75.0
       P1 = 100.0
```

```
      T1 = 300.0
      DPT = 2.5
      W = 180.0
      EF1 = 0.60
      R1O = 0.5
      R1I = 0.25
      A1M = 0.0
      GMA = 1.0
      D = 0.1
      SC = 1.5
      AC = 0.5
      HC = 1.0
      GAP = 0.1
      NSEC = 3
      MATRHO = 500.
      K = 0
      N = 0
**
**
*************************************************************************
****************** BEGIN AERODYNAMIC DESIGN **********************
*************************************************************************
**
***************** calculate nondimensional parameters ********************
**
     CALL NONDIM(DPT,RHO,MF,W,R1O,P1,T1,RGAS,SPSPD,SPDIA,HDCOF,PI,
    $FLFAC,R1I)
**
```

```
********************** estimate blade angles at mean radius *************
**
      CALL MEAN(PI,RGAS,B,MF,P1,T1,DPT,W,EF1,R1O,R1I,A1M,INMED,
     $GMA,SC,AC,HC,RHO,VX1,VX2,V1M,PT1,PT2,RM,UM,B1M,BR1M,WORK,
     $B2M,BR2M,BR2MD,BR1MD,DEL)
**
******************* estimate losses at mean radius ***********************
**
      CALL MLOSS(MF,VX1,VX2,UM,SC,HC,B1M,B2M,WORK,WP,WA,WS,A2M,EF2)
**
******************* output results thus far ******************************
**
      CALL OUT1(MF,DPT,VX1,BR1MD,BR2MD,WP,WA,WS,EF2,SPSPD,SPDIA)
**
**************** start free vortex flow calculations *********************
**
      CALL FVORTX(VX1,VX2,RM,A1M,A2M,W,R1I,R1O,NS,INMED,SC,PI,
     $BR1D,BR2D,BR1,BR2,B1,B2,R,U,A1)
**
******************* output results thus far ******************************
**
      CALL OUT2(R,BR1D,BR2D,NSEC)
**
**************** calculate geometry needed by CADAM **********************
**
      CALL GEOBLD(BR1D,BR2D,BR1,BR2,D,NS,IBR1D,IBR2D,IBR1M,IBR2M,
     $IBR1S,IBR2S,RARC,ST,ED,CT,R)
**
```

```
****************** output results thus far ******************************
**
      CALL OUT3(NS,R,RARC,IBR1D,IBR1M,IBR1S,IBR2D,IBR2M,IBR2S,
     $ST,ED,CT)
**
****************** estimate 3-dimensional losses ************************
**
      CALL LOSS3D(SPDIA,SPSPD,R1I,R1O,RM,EF2,PI,SC,HC,GAP,HDCOF,FLFAC,
     $PTOT,ETOT,CTOT,EF3)
**
****************** output results thus far ******************************
**
      CALL OUT4(PTOT,ETOT,CTOT,EF3)
**
**
*************************************************************************
****************** BEGIN GEOMETRIC MODELING ************************
*************************************************************************
**
****************** start a CADAM drawing ********************************
**
      CALL CADST(DRAWID,USERID,GROUP)
**
****************** begin a CADAM view on drawing ************************
**
      IOPT = 1
      CALL BEGVU(IOPT,IDVU,XYZ,XVCTR,YVCTR,ZVCTR,*7000)
**
```

```
**********************************************************************
********************** model camberlines ******************************
**
************ calculate the geometric coefficients for camberlines ******
**
      DO 6600 M = 1,NS
        DO 6610 J = 1,3
          PTA(J) = ST(M,J)
          PTB(J) = ED(M,J)
          CPT(J) = CT(M,J)
 6610   CONTINUE
        CALL ARCPC(PTA,PTB,CPT,OUT)
        DO 6620 K = 1,4
          DO 6625 J = 1,3
            COARC(M,K,J) = OUT(K,J)
 6625     CONTINUE
 6620   CONTINUE
 6600 CONTINUE
**
************* calculate the points on the camberline ******************
**
     DO 6630 M = 1,NS
       DO 6640 K = 1,4
         DO 6645 KK = 1,3
           OUT(K,KK) = COARC(M,K,KK)
 6645    CONTINUE
 6640  CONTINUE
       DO 6650 J = 1,26
```

```
      IF (J.EQ.1) PU = 0.
      IF (J.EQ.2) PU = 0.005
      IF (J.EQ.3) PU = 0.0075
      IF (J.EQ.4) PU = 0.0125
      IF (J.EQ.5) PU = 0.025
      IF (J.EQ.6) PU = 0.05
      IF (J.EQ.7) PU = 0.075
      IF (J.GT.7) PU = ((FLOAT(J)-6.)*5.0)/100.
      CALL PCRV(OUT,PU,PT)
      DO 6655 K = 1,3
        CAM(M,J,K) = PT(K)
6655    CONTINUE
6650  CONTINUE
6630  CONTINUE
**
****************** draw camberline points ******************************
    DO 6660 M = 1,NS
      DO 6670 J = 1,26
        DO 6675 K = 1,3
          PT(K) = CAM(M,J,K)*100.
6675    CONTINUE
*       CALL PT3D(PT,*7003)
6670  CONTINUE
6660 CONTINUE
**
************************************************************************
****************** model blade cross sections **************************
**
```

```
********** calculate points on blade cross sections **********
**
      DO 6680 M = 1,NS
        TH1 = BR1(M)
        TH2 = BR2(M)
        TMAXFR = 0.1
        CALL COORD(M,TH1,TH2,R,RARC,CAM,CT,ST,ED,TOP,BOT,TMAXFR,NS)
 6680 CONTINUE
**          put call to blade calculation subroutine here
      CALL BLPROP(TOP,BOT,R,AREA,CG,THETA,IMAX,IMIN,IXXO,IYYO,NS,
     $F,BS,TS,F0,CG1,IXX,IYY,MATRHO,W,CSTRSS)
**
********** draw upper surface points **********
**
      DO 6685 M = 1,NS
        DO 6690 J = 1,26
          DO 6695 K = 1,3
            PT(K) = TOP(M,J,K)*100.
 6695       CONTINUE
*         CALL PT3D(PT,*7003)
 6690     CONTINUE
 6685 CONTINUE
**
********** draw lower surface points **********
**
      DO 6700 M = 1,NS
        DO 6705 J = 1,26
          DO 6710 K = 1,3
```

```
            PT(K) = BOT(M,J,K)*100.
 6710       CONTINUE
*       CALL PT3D(PT,*7003)
 6705    CONTINUE
 6700 CONTINUE
**
****** calculate geometric coefficients for cross sectional pc's *******
**
     CALL CURVE(TOP,BOT,CAM,COTOP1,COTOP2,COTOP3,COTOP4,
    $COBOT1,COBOT2,COBOT3,COBOT4,NS)
**
**********************************************************************
******************** model rotated cross sections **********************
**
****************** DRAW rotated blade cross sections********************
**
     NBLADE = INT((R1O*PI)/(SC*D))
     DANG = (2*PI)/FLOAT(NBLADE)
     DO 6111 I = 1,11
     ANGLE = FLOAT(I)*DANG
     CALL ROTLN(TOP,ANGLE,R1I,NS,CIR)
     CALL ROTLN(BOT,ANGLE,R1I,NS,CIR)
 6111 CONTINUE
      IOPT = 2
      NOGOOD = 0
      CALL CADFIL(IOPT,NOGOOD,IDUMMY)
      IF(NOGOOD.NE.1)THEN
      WRITE(NOUT,*) 'ERROR EXISTS IN CADFIL'
```

```
      ENDIF
**
****************** start a CADAM drawing *********************************
**
      DRAWID='BLADE,SURF'
      CALL CADST(DRAWID,USERID,GROUP)
**
****************** begin a CADAM view on drawing *************************
**
      IOPT=1
      CALL BEGVU(IOPT,IDVU,XYZ,XVCTR,YVCTR,ZVCTR,*7000)
**
**
****************** draw surfaces *****************************************
**
     CALL SURF(COTOP1,NS)
     CALL SURF(COTOP2,NS)
     CALL SURF(COTOP3,NS)
     CALL SURF(COTOP4,NS)
**
     CALL SURF(COBOT1,NS)
     CALL SURF(COBOT2,NS)
     CALL SURF(COBOT3,NS)
     CALL SURF(COBOT4,NS)
**
************************** CADCD error messages **************************
**
      GOTO 7100
```

```
7000  WRITE(NOUT,*) 'ERROR EXISTS IN BEGVU'
      GOTO 7200
7001  WRITE(NOUT,*) 'ERROR EXISTS IN CADSP1, camber'
      GOTO 7200
7002  WRITE(NOUT,*) 'ERROR EXISTS IN CADSP1, thickness'
      GOTO 7200
7003  WRITE(NOUT,*) 'ERROR EXISTS IN 3D POINT'
      GOTO 7200
7100  IOPT=2
      NOGOOD=0
      CALL CADFIL(IOPT,NOGOOD,IDUMMY)
      IF(NOGOOD.NE.1)THEN
      WRITE(NOUT,*) 'ERROR EXISTS IN CADFIL'
      ENDIF
7200  RETURN
      END
**
**
**********************************************************************
**********************************************************************
******************** END OF CONTROL SUBROUTINE ********************
**********************************************************************
**********************************************************************
**
**
**
**
**
```

```
**
**
**
**********************************************************************
**********************************************************************
**************** SUBROUTINES  CALLED BY CONTRL **********************
**********************************************************************
**********************************************************************
**
**
**
**********************************************************************
*
*    Subroutine:  NONDIM
*
*    Purpose:     compute the nondimensional parameters describing
*                 aerodynamic operation
*
**********************************************************************
      SUBROUTINE NONDIM(DPT,RHO,MF,W,R1O,P1,T1,RGAS,SPSPD,SPDIA,HDCOF
     $,PI,FLFAC,R1I)
      REAL DPT,RHO,MF,W,R1O,P1,T1,RGAS,PI
      REAL HDTOT,VFLOW,SPSPD,SPDIA,HDCOF,FLFAC
      G=9.81
      RHO=P1/(RGAS*T1)
      HDTOT=(DPT*1000)/(RHO*G)
      VFLOW=MF/RHO
      SPSPD=(W*VFLOW**0.5)/(G*HDTOT)**0.75
```

```
      SPDIA = ((2.*R1O)*(HDTOT*G)**0.25)/VFLOW**0.5
      HDCOF = (G*HDTOT)/((W*2.*R1O)**2)
      FLFAC = 8./(PI*(1-(R1I/R1O)**2)*SPSPD*SPDIA)
      RETURN
      END
**
**
**
**
**
*************************************************************************
*
*     Subroutine:  MEAN
*
*     Purpose:     calculate the blade and flow angles at the mean
*                  radius for an axial-flow fan blade
*
*************************************************************************
      SUBROUTINE MEAN(PI,RGAS,B,MF,P1,T1,DPT,W,EF1,R1O,R1I,A1M,INMED,
     $GMA,SC,AC,HC,RHO,VX1,VX2,V1M,PT1,PT2,RM,UM,B1M,BR1M,WORK,
     $B2M,BR2M,BR2MD,BR1MD,DEL)
      REAL PI,RGAS,B,MF,P1,T1,DPT,W,EF1,R1O,R1I,A1M,GMA,SC,AC,HC
      REAL RHO,VX1,VX2,V1M,PT1,PT2,RM,UM,B1M,BR1M,WORK,B2M,INMED
      REAL BR2M,BR2MD,BR1MD,DEL
        VX1 = MF/((R1O**2-R1I**2)*PI*RHO*B)
        VX2 = VX1
        V1M = VX1/COS(A1M)
        PT1 = ((RHO*(V1M**2.))/2000) + P1
```

```
      PT2 = PT1 + DPT
      RM = ((R1O**2 + R1I**2)/2)**0.5
      UM = RM*W
      B1M = ATAN((UM/VX1)-TAN(A1M))
      BR1M = B1M
      WORK = (PT1-PT2)/(RHO*EF1)
      B2M = ATAN((((WORK*1000./(GMA*UM)) + UM)/VX1)-TAN(A1M))
      INMED = (0.23*4*(AC**2)) + (B2M/500.0)
      BR2M = ((BR1M*INMED*(SC**0.5))-B2M)/(((SC**0.5)*INMED)-1.)
      BR2MD = BR2M*(180./PI)
      BR1MD = BR1M*(180./PI)
      DEL = B2M-BR2M
    RETURN
    END
**
**
**
*************************************************************************
*
*    Subroutine:  MLOSS
*
*    Purpose:     estimate the blade performance based on
*                 meanline geometry
*
*************************************************************************
    SUBROUTINE MLOSS(MF,VX1,VX2,UM,SC,HC,B1M,B2M,WORK,WP,WA,WS,A2M
   $,EF2)
    REAL VX1,VX2,UM,SC,HC,B1M,B2M,EF2,WORK,WP,WA,WS,EF,MF
```

```
      DR = (COS(B2M)/COS(B1M))*(((TAN(B1M)-TAN(B2M))*
     $(COS(B1M)**2)*SC*0.61) + 1.12)
      TH2C = 0.004/(1.-(1.17*ALOG(DR)))
      WP = ((COS(B1M)**2)/(COS(B2M)**3))*TH2C*2.0/SC
      BM = 0.5*(TAN(B1M) + TAN(B2M))
      WA = ((COS(B1M)**2)/(COS(BM)**3))*0.02/HC
      WS = ((TAN(B1M)-TAN(B2M))**2)*(COS(B1M)**2)*SC*0.08/COS(BM)
      WROT = WP + WA + WS
      WSTAT = WROT
      W1M = VX1/COS(B1M)
      A2M = ATAN((UM-(VX2*TAN(B2M)))/VX2)
      V2M = VX2/COS(A2M)
      EF2 = 1-((((W1M**2)*WROT) + ((V2M**2)*WSTAT))/(-2000.*WORK))
      RETURN
      END
**
**
**
*************************************************************************
*
*     Subroutine:  FVORTX
*
*     Purpose:     compute flow and blade angles of an axial-flow
*                  based on free vortex flow distribution
*
*************************************************************************
      SUBROUTINE FVORTX(VX1,VX2,RM,A1M,A2M,W,R1I,R1O,NS,INMED,SC,PI,
```

```
     $BR1D,BR2D,BR1,BR2,B1,B2,R,U,A1)
*    REAL R(10),A1(10),U(10),BR1(10),BR2(10),B2(10),BR1D(10),BR2D(10)
     REAL R(NS), A1(NS), U(NS), BR1(NS), BR2(NS), B2(NS)
     REAL BR1D(NS), BR2D(NS)
     REAL INMED
     C1 = RM*TAN(A1M)
     C2 = RM*VX2*TAN(A2M)
**
**        calculate angles, hub to mean radius
**
     DO 1000 I = 1,NS
     R(1) = R1I
     IF (I.EQ.1) GOTO 700
     R(I) = ((((R1O**2-R1I**2)/8.) + (R(I-1)**2))**0.5)
 700 A1(I) = ATAN(C1/R(I))
     U(I) = R(I)*W
     BR1(I) = (U(I)/VX1)-(TAN(A1(I)))
     VT2 = C2/R(I)
     B2(I) = ATAN((U(I)-VT2)/VX2)
     BR2(I) = ((BR1(I)*INMED*(SC**0.5))-B2(I))/(((SC**0.5)*INMED)-1.)
     BR1D(I) = BR1(I)*180/PI
     BR2D(I) = BR2(I)*180/PI
1000 CONTINUE
**
**        calculate angles, mean radius to tip
**
     I1 = ((NS-1)/2) + 1
     I2 = NS
```

```
      DO 2000 I = I1,I2
      R(I) = ((((R1O**2-R1I**2)/8.) + (R(I-1)**2))**0.5)
      A1(I) = ATAN(C1/R(I))
      U(I) = R(I)*W
      BR1(I) = (U(I)/VX1)-(TAN(A1(I)))
      VT2 = C2/R(I)
      B2(I) = ATAN((U(I)-VT2)/VX2)
      BR2(I) = ((BR1(I)*INMED*(SC**0.5))-B2(I))/(((SC**0.5)*INMED)-1.)
      BR1D(I) = BR1(I)*180./PI
      BR2D(I) = BR2(I)*180./PI
 2000 CONTINUE
      RETURN
      END
**
**
**
*************************************************************************
*
*     Subroutine:  GEOBLD
*
*     Purpose:     compute the geometry required to model the blade
*                  cross sections using CADAM
*
*************************************************************************
      SUBROUTINE GEOBLD(BR1D,BR2D,BR1,BR2,D,NS,IBR1D,IBR2D,IBR1M,IBR2M,
     $IBR1S,IBR2S,RARC,ST,ED,CT,R)
      INTEGER IBR1D(NS),IBR1M(NS),IBR1S(NS)
      INTEGER IBR2D(NS),IBR2M(NS),IBR2S(NS)
```

```
      REAL BR1(NS),BR2(NS)
      REAL BR1D(NS),BR2D(NS)
      REAL RARC(NS),R(NS)
      REAL ST(NS,3),ED(NS,3),CT(NS,3)
        DO 5000 I = 1,NS
        IBR1D(I) = INT(BR1D(I))
        IBR2D(I) = INT(BR2D(I))
        REM1 = BR1D(I)-FLOAT(IBR1D(I))
        REM2 = BR2D(I)-FLOAT(IBR2D(I))
        IBR1M(I) = INT(REM1*60.)
        IBR2M(I) = INT(REM2*60.)
        REM1 = (REM1*60.)-FLOAT(IBR1M(I))
        REM2 = (REM2*60.)-FLOAT(IBR2M(I))
        IBR1S(I) = INT(REM1*60.)
        IBR2S(I) = INT(REM2*60.)
        RARC(I) = D/(SIN(BR1(I)) + SIN(-1*BR2(I)))
        ST(I,1) = 0.0
        ST(I,2) = 0.0
        ST(I,3) = R(I)
        CT(I,1) = ST(I,1) + (RARC(I)*SIN(BR1(I)))
        CT(I,2) = ST(I,2)-(RARC(I)*COS(BR1(I)))
        CT(I,3) = R(I)
        ED(I,1) = D
        ED(I,2) = CT(I,2) + (RARC(I)*COS(BR2(I)))
        ED(I,3) = R(I)
5000    CONTINUE
        RETURN
```

```
      END
**
**
*************************************************************************
*
*    Subroutine:  LOSS3D
*
*    Purpose:     estimate the losses in a three-dimensional axial-flow
*                 blade based on work by Balje
*
*************************************************************************
      SUBROUTINE LOSS3D(SPDIA,SPSPD,R1I,R1O,RM,EF2,PI,SC,HC,GAP,
     $HDCOF,FLFAC,PTOT,ETOT,CTOT,EF3)
      REAL SPDIA,SPSPD,R1I,R1O,EF2,PI,SC
      REAL BL(3),R(3),MVEC(3),CIR(3),EQDIF(3),MOMTHK(3)
      REAL PLOSS(3),ELOSS(3),CLOSS(3)
      H = 0.3861
      R(1) = R1I
      R(2) = RM
      R(3) = R1O
      DO 100 I = 1,3
      BL(I) = (PI*SPDIA*(1-(R1I/R1O)**2))/((R(I)/R1O)*EF2*SPSPD)
      X1 = 1./((R(I)/R1O)*EF2*SPSPD*4.)
      X2 = (SPSPD*SPDIA**2)/(8.*(R1O/R(I)))
      MVEC(I) = PI*SPDIA*(1-(R1I/R1O)**2)*(X1-X2)
      CIR(I) = (2.*BL(I))/(SC*(1 + (MVEC(I)-(BL(I)/2)**2)))
      EQDIF(I) = 1.09 + (0.3*CIR(I))
      MOMTHK(I) = (0.013/(2.62-EQDIF(I)))-0.004
```

```
      X1 = 1.286*(H**2)*SC*MOMTHK(I)*(1 + (MVEC(I) + (BL(I)/2.))**2)**1.5
      X2 = (1-(H*SC*MOMTHK(I)*(1-(MVEC(I) + (BL(I)/2.))**2)**0.5))**3
      PLOSS(I) = X1/X2
      ELOSS(I) = (PLOSS(I)*HC)/SC
      X1 = (GAP*HC)**0.5
      X2 = 0.19*BL(I)*(1 + (MVEC(I) + (BL(I)/2.))**2)*0.01
      X3 = (1 + MVEC(I)**2)**0.5
      CLOSS(I) = X2/(X1*X3)
  100 CONTINUE
      PTOT = (PLOSS(1) + PLOSS(2) + PLOSS(3))/3.
      ETOT = (ELOSS(1) + ELOSS(2) + ELOSS(3))/3.
      CTOT = (CLOSS(1) + CLOSS(2) + CLOSS(3))/3.
      EF3 = 1.-((FLFAC**2*(PTOT + ETOT + CTOT))/(2.*HDCOF))
      RETURN
      END
**
**
*************************************************************************
*
*     Subroutine:  BLPROP
*
*     Purpose:     compute the cross-sectional properties at various
*                  points along the blade length
*
*************************************************************************
      SUBROUTINE BLPROP(TOP,BOT,R,AREA,CG,THETA,IMAX,IMIN,IXX0,IYY0,NS,
     $F,BS,TS,F0,CG1,IXX,IYY,MATRHO,W,CSTRSS)
      REAL TOP(NS,26,3),BOT(NS,26,3),R(NS)
```

```
      REAL AREA(NS),CG1(NS,3),CG(NS,3),F(NS),IXX(NS),IYY(NS)
      REAL IXX0(NS),IYY0(NS)
      REAL F0(NS),IMAX(NS),IMIN(NS),THETA(NS),TS(NS,26,3),BS(NS,26,3)
      REAL CSTRSS(NS)
      G=9.81
**
*************************** zero arrays ********************************
**
      DO 3 M=1,NS
        DO 5 K=1,3
          AREA(M)=0.
          CG1(M,K)=0.
          CG(M,K)=0.
          THETA(M)=0.
          F(M)=0.
          IXX(M)=0.
          IYY(M)=0.
          CSTRSS(M)=0.
 5      CONTINUE
 3    CONTINUE
**
******** calculate area of each cross-section ************************
**
      DO 10 M=1,NS
        DO 20 L=1,25
          AREA(M)=AREA(M)+(0.5*(TOP(M,L+1,1)-TOP(M,L,1))*(TOP(M,L,2)
     $      +TOP(M,L+1,2)-BOT(M,L,2)-BOT(M,L+1,2)))
20      CONTINUE
```

```
 10   CONTINUE
**
********* estimate initial center of gravity for each cross-section ****
**
      DO 30 M = 1,NS
        DO 40 L = 1,25
          CG1(M,1) = CG1(M,1) + ((1./(4.*AREA(M)))*((TOP(M,L + 1,1)**2-
     $     TOP(M,L,1)**2)*(TOP(M,L + 1,2) + TOP(M,L,2)-BOT(M,L + 1,2)-
     $     BOT(M,L,2))))
          CG1(M,2) = CG1(M,2) + ((1./(8.*AREA(M))*((TOP(M,L + 1,1)-
     $     TOP(M,L,1))*((TOP(M,L + 1,2) + TOP(M,L,2))**2-(BOT(M,L + 1,2)
     $     + BOT(M,L,2))**2))))
          CG1(M,3) = R(M)
 40     CONTINUE
 30   CONTINUE
**
***************** shift axes ********************************************
**
      DO 50 M = 1,NS
        DO 60 L = 1,26
          DO 65 K = 1,3
            TS(M,L,K) = TOP(M,L,K)-CG1(M,K)
            BS(M,L,K) = BOT(M,L,K)-CG1(M,K)
 65       CONTINUE
 60     CONTINUE
 50   CONTINUE
**
```

```
********** recalculate center of gravities **********
**
      DO 70 M = 1,NS
        DO 80 L = 1,25
          CG(M,1) = CG(M,1) + ((1./(4.*AREA(M)))*((TOP(M,L + 1,1)**2-
     $     TOP(M,L,1)**2)*(TOP(M,L + 1,2) + TOP(M,L,2)-BOT(M,L + 1,2)-
     $     BOT(M,L,2))))
          CG(M,2) = CG(M,2) + ((1./(8.*AREA(M))*((TOP(M,L + 1,1)-
     $     TOP(M,L,1))*((TOP(M,L + 1,2) + TOP(M,L,2))**2-(BOT(M,L + 1,2)
     $      + BOT(M,L,2))**2))))
          CG(M,3) = R(M)
 80     CONTINUE
 70   CONTINUE
**
********** calculate moments of inertia **********
**
      DO 90 M = 1,NS
        DO 100 L = 1,25
        F(M) = F(M) +
     $  ((1./16.)*(TOP(M,L + 1,1)**2-TOP(M,L,1)**2)*((TOP(M,L + 1,2)
     $   + TOP(M,L,2))**2-(BOT(M,L + 1,2) + BOT(M,L,2))**2))
        IXX(M) = IXX(M) +
     $  ((1./24.)*(TOP(M,L + 1,1)-TOP(M,L,1))*((TOP(M,L + 1,2)
     $   + TOP(M,L,2))**3-(BOT(M,L + 1,2) + BOT(M,L,2))**3))
        IYY(M) = IYY(M) + ((TOP(M,L,1)-TOP(M,L + 1,1))*(TOP(M,L + 1,2) +
     $TOP(M,L,2)-BOT(M,L + 1,2)-BOT(M,L,2))*((((TOP(M,L,1) + TOP(M,L + 1,1))**
     $2)/8.) + (((TOP(M,L,1)-TOP(M,L + 1,1))**2)/24.)))
 100     CONTINUE
```

```
 90   CONTINUE
**
**** use parallel-axis theorem to transfer properties to a set of
**** parallel axes passing through the centers of gravity ***************
**
      DO 110 M = 1,NS
        IXX0(M) = IXX(M)-(AREA(M)*CG(M,2)**2)
        IYY0(M) = IYY(M)-(AREA(M)*CG(M,1)**2)
        F0(M) = F(M)-(CG(M,1)*CG(M,2)*AREA(M))
 110  CONTINUE
**
************ calculate minimum and maximum moments of inertia ************
**
      DO 120 M = 1,NS
        THETA(M) = 0.5*ATAN((2.*F0(M))/(IYY0(M)-IXX0(M)))
        IMAX(M) = 0.5*(IXX0(M) + IYY0(M) + ((IYY0(M)-IXX0(M))*(1./COS(2.*
     $  THETA(M)))))
        IMIN(M) = 0.5*(IXX0(M) + IYY0(M)-((IYY0(M)-IXX0(M))*(1./COS(2.
     $  *THETA(M)))))
 120  CONTINUE
**
******************** calculate centrifugal stresses **********************
**
      DO 200 M = 1,NS-1
        DO 250 L = 1,M
          CSTRSS(M) = CSTRSS(M) + (((MATRHO*W**2)/(AREA(M)*G))*0.5*
     $((AREA(L)*R(L)) + (AREA(L + 1)*R(L + 1)))*(R(L + 1)-R(L)))
 250    CONTINUE
```

```
200  CONTINUE
     CSTRSS(NS) = 0.
     RETURN
     END
**
**
*************************************************************************
*
*       SUBROUTINE ARCPC
*
*       Purpose:  compute the geometric coefficients of a pc
*                    curve that approximates a circular arc of given
*                 radius and center point
*
*                 P0(3):  input array of coordinates at u = 0
*                 P1(3):  input array of coordinates at u = 1
*                 CP(3):  input array of coordinates of center point
*                 CO(4,3):  output array of geometric coefficients
*
*************************************************************************
     SUBROUTINE ARCPC(P0,P1,CP,CO)
     REAL P0(3),P1(3),CP(3),CO(4,3),DP10(3),DPC0(3),DPC1(3)
     REAL MAG1,MAG2,MAG10,MAGC0
     DO 10 I = 1,3
     DP10(I) = P1(I)-P0(I)
     DPC0(I) = CP(I)-P0(I)
     DPC1(I) = CP(I)-P1(I)
10   CONTINUE
```

```
      CALL VMAG(DP10,MAG10)
      CALL VMAG(DPC0,MAGC0)
       X = MAG10/(2.*MAGC0)
      THETA = ASIN(MAG10/(2.*MAGC0))
      CONS = 2./(COS(THETA)*(1 + COS(THETA)))
      DO 20 I = 1,3
      CO(1,I) = P0(I)
      CO(2,I) = P1(I)
      CO(3,I) = CONS*(DP10(I)-(2*SIN(THETA)**2*DPC0(I)))
      CO(4,I) = CONS*(DP10(I) + (2*SIN(THETA)**2*DPC1(I)))
   20 CONTINUE
      RETURN
      END
*************************************************************************
*
*       SUBROUTINE PCRV
*
*       Purpose:  compute x,y,z coordinates of a point on PC
*                 curve for a given u. (geometric form)
*
*             CI(4,3):  input array of geometric coefficients
*             U:        input value of parameter u
*             P(3):     output array of x,y,z coordinates
*
*************************************************************************
      SUBROUTINE PCRV(CI,U,P)
      DIMENSION CI(4,3),P(3),F(4)
      F(1) = (2.*U**3)-(3.*U**2) + 1
```

```
      F(2) = (-2.*U**3) + (3.*U**2)
      F(3) = (U**3)-(2.*U**2) + U
      F(4) = (U**3)-(U**2)
      DO 10 I = 1,3
      P(I) = (F(1)*CI(1,I)) + (F(2)*CI(2,I)) + (F(3)*CI(3,I)) + (F(4)*CI(4,I))
   10 CONTINUE
      RETURN
      END
*
*     SUBROUTINE VMAG
*
*     Purpose:  Compute magnitude of vector
*
*               V(3): input array of component P1
*               MAG: magnitude of input array
*
      SUBROUTINE VMAG(V,MAG)
      REAL V(3),MAG
      MAG = (V(1)**2 + V(2)**2 + V(3)**2)**0.5
      RETURN
      END
*
*************************************************************************
*
*     SUBROUTINE COORD
*
*     Purpose:  calculate coordinates of points on upper and lower
*               surfaces of an NACA 65-series airfoil
```

```
*
*
**********************************************************************
      SUBROUTINE COORD(M,TH1,TH2,R,RARC,CAM,CT,ST,ED,TOP,BOT,TMAXFR,NS)
      REAL CAM(NS,26,3),BOT(NS,26,3),TOP(NS,26,3),R(NS),RARC(NS)
      REAL CT(NS,3),ST(NS,3),ED(NS,3),THK(26)
      DATA THK/0.,0.00752,0.0089,0.01124,0.01571,0.02222,0.02709
     $,0.03111,0.03746,0.04218,0.0457,0.04824,0.04982,0.05057
     $,0.05029,0.0487,0.0456,0.04151,0.03627,0.03038,0.02451,0.01847
     $,0.01251,0.00749,0.00354,0.0015/
      PI = 3.14159
      ARCLN = RARC(M)*(TH1-TH2)
      TRAT = TMAXFR/0.1
      DO 100 J = 1,26
        IF (J.EQ.1) PU = 0.
        IF (J.EQ.2) PU = 0.005
        IF (J.EQ.3) PU = 0.0075
        IF (J.EQ.4) PU = 0.0125
        IF (J.EQ.5) PU = 0.025
        IF (J.EQ.6) PU = 0.05
        IF (J.EQ.7) PU = 0.075
        IF (J.GT.7) PU = ((FLOAT(J)-6.)*5.)/100.
        ANGLE = (PI/2.)-TH1 + (PU*(TH1-TH2))
      BOT(M,J,1) = CT(M,1)-((RARC(M)-(THK(J)*ARCLN*TRAT))*
     $COS(ANGLE))
      TOP(M,J,1) = CT(M,1)-((RARC(M) + (THK(J)*ARCLN*TRAT))*
     $COS(ANGLE))
      BOT(M,J,2) = CT(M,2) + ((RARC(M)-(THK(J)*ARCLN*TRAT))*
```

```
     $SIN(ANGLE))
      TOP(M,J,2) = CT(M,2) + ((RARC(M) + (THK(J)*ARCLN*TRAT))*
     $SIN(ANGLE))
      BOT(M,J,3) = R(M)
      TOP(M,J,3) = R(M)
  100 CONTINUE
      RETURN
      END
*************************************************************************
*
*       SUBROUTINE CURVE
*
*       Purpose: Convert 4-point form parametric cubics to
*                geometric form
*
*************************************************************************
      SUBROUTINE CURVE(TOP,BOT,CAM,COTOP1,COTOP2,COTOP3,COTOP4,
     $COBOT1,COBOT2,COBOT3,COBOT4,NS)
      REAL PTT(4,3),PTB(4,3),COTOP1(NS,4,3),COTOP2(NS,4,3),
     $COTOP3(NS,4,3),COTOP4(NS,4,3),COBOT1(NS,4,3),COBOT2(NS,4,3),
     $COBOT3(NS,4,3),COBOT4(NS,4,3),COT(4,3),COB(4,3),
     $TOP(NS,26,3),BOT(NS,26,3),CAM(NS,26,3)
**
      DO 100 M = 1,NS
        DO 125 J = 1,4
          DO 130 K = 1,3
            PTT(J,K) = TOP(M,J,K)
            PTB(J,K) = BOT(M,J,K)
```

```
130       CONTINUE
125     CONTINUE
        CALL FPTOG(PTT,COT)
        CALL FPTOG(PTB,COB)
        DO 135 J = 1,4
          DO 140 K = 1,3
            COTOP1(M,J,K) = COT(J,K)
            COBOT1(M,J,K) = COB(J,K)
140       CONTINUE
135     CONTINUE
******
**
        DO 145 J = 1,4
          DO 150 K = 1,3
            INC = (2*J) + 2
            PTT(J,K) = TOP(M,INC,K)
            PTB(J,K) = BOT(M,INC,K)
150       CONTINUE
145     CONTINUE
        CALL FPTOG(PTT,COT)
        CALL FPTOG(PTB,COB)
        DO 155 J = 1,4
          DO 160 K = 1,3
            COTOP2(M,J,K) = COT(J,K)
            COBOT2(M,J,K) = COB(J,K)
160       CONTINUE
155     CONTINUE
**
```

```
      DO 165 J = 1,4
        DO 170 K = 1,3
          IF (J.EQ.1) INC = 10
          IF (J.EQ.2) INC = 12
          IF (J.EQ.3) INC = 16
          IF (J.EQ.4) INC = 19
          PTT(J,K) = TOP(M,INC,K)
          PTB(J,K) = BOT(M,INC,K)
170       CONTINUE
165     CONTINUE
      CALL FPTOG(PTT,COT)
      CALL FPTOG(PTB,COB)
      DO 175 J = 1,4
        DO 180 K = 1,3
          COTOP3(M,J,K) = COT(J,K)
          COBOT3(M,J,K) = COB(J,K)
180       CONTINUE
175     CONTINUE
**
      DO 185 J = 1,4
        DO 190 K = 1,3
          IF (J.EQ.1) INC = 19
          IF (J.EQ.2) INC = 22
          IF (J.EQ.3) INC = 24
          IF (J.EQ.4) INC = 26
          PTT(J,K) = TOP(M,INC,K)
          PTB(J,K) = BOT(M,INC,K)
190       CONTINUE
```

```
185    CONTINUE
       CALL FPTOG(PTT,COT)
       CALL FPTOG(PTB,COB)
       DO 195 J = 1,4
         DO 200 K = 1,3
           COTOP4(M,J,K) = COT(J,K)
           COBOT4(M,J,K) = COB(J,K)
200      CONTINUE
195    CONTINUE
100  CONTINUE
     RETURN
     END
*************************************************************************
*
*       SUBROUTINE FPTOG
*
*       Purpose:  Compute geometric coefficients of a pc, given
*                 4 points on the pc
*
*             P(4,3):  input array of 4 pts. on pc
*             CO(4,3:  output array of geometric coefficients
*
*       NOTE:    THIS ROUTINE ASSUMES EQUAL DIVISIONS OF U
*
*************************************************************************
     SUBROUTINE FPTOG(P,CO)
     REAL P(4,3),CO(4,3),K(4,4)
     K(1,1) = 1.0
```

```
      K(1,2) = 0.0
      K(1,3) = 0.0
      K(1,4) = 0.0
      K(2,1) = 0.0
      K(2,2) = 0.0
      K(2,3) = 0.0
      K(2,4) = 1.0
      K(3,1) = (-11./2.)
      K(3,2) = (9.0)
      K(3,3) = (-9./2.)
      K(3,4) = 1.0
      K(4,1) = (-1.0)
      K(4,2) = (9./2.)
      K(4,3) = (-9.0)
      K(4,4) = (11./2.)
      DO 40 J = 1,3
      DO 50 I = 1,4
      CO(I,J) = (K(I,1)*P(1,J)) + (K(I,2)*P(2,J)) + (K(I,3)*P(3,J)) +
     $(K(I,4)*P(4,J))
   50 CONTINUE
   40 CONTINUE
      RETURN
      END
*************************************************************************
*
*       SUBROUTINE REPARM
*
*       Purpose:  reparameterize cross-sectional pc's
```

```
*
*************************************************************************
      SUBROUTINE REPARM(COTOP2,COTOP3,COTOP4,COBOT1,COBOT2,COBOT3)
      REAL COTOP2(9,4,3),COTOP3(9,4,3),COTOP4(9,4,3)
     $,COBOT2(9,4,3),COBOT3(9,4,3),COBOT4(9,4,3)
     $,CIT(4,3),CIB(4,3),U(4),COT(4,3),COB(4,3)
      U(1)=0.
      U(2)=1.
      DO 100 M=1,9
        DO 200 J=1,4
          DO 210 K=1,3
            CIT(J,K)=COTOP2(M,J,K)
            CIB(J,K)=COBOT2(M,J,K)
210       CONTINUE
200     CONTINUE
        U(3)=1.
        U(4)=2.
        CALL RPRCRV(CIT,U,COT)
        CALL RPRCRV(CIB,U,COB)
        DO 220 J=1,4
          DO 230 K=1,3
            COTOP2(M,J,K)=COT(J,K)
            COBOT2(M,J,K)=COB(J,K)
230       CONTINUE
220     CONTINUE
100     CONTINUE
**
      DO 300 M=1,9
```

```
      DO 400 J = 1,4
        DO 410 K = 1,3
          CIT(J,K) = COTOP3(M,J,K)
          CIB(J,K) = COBOT3(M,J,K)
410       CONTINUE
400     CONTINUE
      U(3) = 2.
      U(4) = 3.
      CALL RPRCRV(CIT,U,COT)
      CALL RPRCRV(CIB,U,COB)
      DO 420 J = 1,4
        DO 430 K = 1,3
          COTOP3(M,J,K) = COT(J,K)
          COBOT3(M,J,K) = COB(J,K)
430       CONTINUE
420     CONTINUE
300     CONTINUE
**
    DO 500 M = 1,9
      DO 600 J = 1,4
        DO 610 K = 1,3
          CIT(J,K) = COTOP4(M,J,K)
          CIB(J,K) = COBOT4(M,J,K)
610       CONTINUE
600     CONTINUE
      U(3) = 3.
      U(4) = 4.
      CALL RPRCRV(CIT,U,COT)
```

```
      CALL RPRCRV(CIB,U,COB)
      DO 620 J = 1,4
        DO 630 K = 1,3
          COTOP4(M,J,K) = COT(J,K)
          COBOT4(M,J,K) = COB(J,K)
630       CONTINUE
620     CONTINUE
500     CONTINUE
**
      RETURN
      END
*************************************************************************
*
*       SUBROUTINE RPRCRV
*
*       Purpose:  reparametrize a pc curve
*
*               CI(4,3):  input array of geometric coefficients
*               U(4):     input array of u, U(1) to U(2)  old
*                                  U(3) to U(4)new
*               CO(4,3):  output array of reparam. geo coefficients
*
*************************************************************************
      SUBROUTINE RPRCRV(CI,U,CO)
      REAL CI(4,3),U(4),CO(4,3)
      DELTA = (U(2)-U(1))/(U(4)-U(3))
      DO 10 I = 1,3
```

```
      CO(1,I) = CI(1,I)
A
      CO(2,I) = CI(2,I)
      CO(3,I) = CI(3,I)*DELTA
      CO(4,I) = CI(4,I)*DELTA
   10 CONTINUE
      RETURN
      END
************************************************************************
*
*       SUBROUTINE CSEC
*
*       Purpose: draw cross-sectional pc's for upper and
*                lower surfaces
*
************************************************************************
      SUBROUTINE CSEC(COTOP1,COTOP2,COTOP3,COTOP4,COBOT1,COBOT2,COBOT3,
     $COBOT4,NS)
      REAL COTOP1(NS,4,3),COTOP2(NS,4,3),COTOP3(NS,4,3),COTOP4(NS,4,3)
     $,COBOT1(NS,4,3),COBOT2(NS,4,3),COBOT3(NS,4,3),COBOT4(NS,4,3),
     $PT(11,3),PB(11,3),PTOP1(3),PBOT1(3),PTOP2(3),PBOT2(3)
     $,CIT(4,3),CIB(4,3)
      NSEG = 10
**
      DO 100 M = 1,9
        DO 110 K = 1,4
          DO 120 J = 1,3
            CIT(K,J) = COTOP1(M,K,J)
```

```
          CIB(K,J) = COBOT1(M,K,J)
120       CONTINUE
110     CONTINUE
        DO 130 N = 1,NSEG + 1
          U = (FLOAT(N)/FLOAT(NSEG))-(1./FLOAT(NSEG))
          CALL PCRV(CIT,U,PTOP1)
          CALL PCRV(CIB,U,PBOT1)
          DO 135 J = 1,3
            PT(N,J) = PTOP1(J)
            PB(N,J) = PBOT1(J)
135       CONTINUE
130     CONTINUE
        DO 140 N = 1,NSEG
          DO 145 J = 1,3
            PTOP1(J) = PT(N,J)*100.
            PTOP2(J) = PT(N + 1,J)*100.
            PBOT1(J) = PB(N,J)*100.
            PBOT2(J) = PB(N + 1,J)*100.
145       CONTINUE
        WRITE(3,*) N,PTOP1(1),PTOP1(2),PTOP1(3)
        WRITE(3,*) N,PTOP2(1),PTOP2(2),PTOP2(3)
        WRITE(3,*) N,PBOT1(1),PBOT1(2),PBOT1(3)
        WRITE(3,*) N,PBOT2(1),PBOT2(2),PBOT2(3)
        CALL LINE3D(PTOP1,PTOP2,*9000)
        CALL LINE3D(PBOT1,PBOT2,*9000)
140   CONTINUE
100 CONTINUE
```

```
      WRITE(*,*) 'one'
**
      DO 200 M = 1,9
        DO 210 K = 1,4
          DO 220 J = 1,3
            CIT(K,J) = COTOP2(M,K,J)
            CIB(K,J) = COBOT2(M,K,J)
220       CONTINUE
210     CONTINUE
        DO 230 N = 1,NSEG + 1
          U = (FLOAT(N)/FLOAT(NSEG))-(1./FLOAT(NSEG))
          CALL PCRV(CIT,U,PTOP1)
          CALL PCRV(CIB,U,PBOT1)
          DO 235 J = 1,3
            PT(N,J) = PTOP1(J)
            PB(N,J) = PBOT1(J)
235       CONTINUE
230     CONTINUE
        DO 240 N = 1,NSEG
          DO 245 J = 1,3
            PTOP1(J) = PT(N,J)*100.
            PTOP2(J) = PT(N + 1,J)*100.
            PBOT1(J) = PB(N,J)*100.
            PBOT2(J) = PB(N + 1,J)*100.
245       CONTINUE
        CALL LINE3D(PTOP1,PTOP2,*9000)
        CALL LINE3D(PBOT1,PBOT2,*9000)
240   CONTINUE
```

```
200  CONTINUE
     WRITE(*,*) 'two'
**
     DO 300 M = 1,9
       DO 310 K = 1,4
         DO 320 J = 1,3
           CIT(K,J) = COTOP3(M,K,J)
           CIB(K,J) = COBOT3(M,K,J)
320      CONTINUE
310    CONTINUE
       DO 330 N = 1,NSEG + 1
         U = (FLOAT(N)/FLOAT(NSEG))-(1./FLOAT(NSEG))
         CALL PCRV(CIT,U,PTOP1)
         CALL PCRV(CIB,U,PBOT1)
         DO 335 J = 1,3
           PT(N,J) = PTOP1(J)
           PB(N,J) = PBOT1(J)
335      CONTINUE
330    CONTINUE
       DO 340 N = 1,NSEG
         DO 345 J = 1,3
           PTOP1(J) = PT(N,J)*100.
           PTOP2(J) = PT(N + 1,J)*100.
           PBOT1(J) = PB(N,J)*100.
           PBOT2(J) = PB(N + 1,J)*100.
345      CONTINUE
       CALL LINE3D(PTOP1,PTOP2,*9000)
       CALL LINE3D(PBOT1,PBOT2,*9000)
```

```
340  CONTINUE
300  CONTINUE
     WRITE(*,*) 'three'
**
     DO 400 M = 1,9
       DO 410 K = 1,4
         DO 420 J = 1,3
           CIT(K,J) = COTOP4(M,K,J)
           CIB(K,J) = COBOT4(M,K,J)
420       CONTINUE
410    CONTINUE
       DO 430 N = 1,NSEG + 1
         U = (FLOAT(N)/FLOAT(NSEG))-(1./FLOAT(NSEG))
         CALL PCRV(CIT,U,PTOP1)
         CALL PCRV(CIB,U,PBOT1)
         DO 435 J = 1,3
           PT(N,J) = PTOP1(J)
           PB(N,J) = PBOT1(J)
435       CONTINUE
430    CONTINUE
       DO 440 N = 1,NSEG
         DO 445 J = 1,3
           PTOP1(J) = PT(N,J)*100.
           PTOP2(J) = PT(N + 1,J)*100.
           PBOT1(J) = PB(N,J)*100.
           PBOT2(J) = PB(N + 1,J)*100.
445       CONTINUE
       CALL LINE3D(PTOP1,PTOP2,*9000)
```

```
      CALL LINE3D(PBOT1,PBOT2,*9000)
440   CONTINUE
400   CONTINUE
      WRITE(*,*) 'four'
      GOTO 9990
9000  WRITE(*,*) 'ERROR IN LINE3D,CSEC'
      STOP
9990  RETURN
      END
*************************************************************************
*
*       SUBROUTINE SURF
*
*       Purpose:  draw surfaces on blade
*
*************************************************************************
      SUBROUTINE SURF(CO,NS)
      REAL CO(NS,4,3),PT(5,4,4),CI(4,4,3),PT1(3),PT2(3)
      DO 100 M=1,NS-1
        DO 110 J=1,3
        CI(1,1,J)=CO(M,1,J)
        CI(2,1,J)=CO(M,2,J)
        CI(1,2,J)=CO(M+1,1,J)
        CI(2,2,J)=CO(M+1,2,J)
**
        CI(3,1,J)=CO(M,3,J)
        CI(4,1,J)=CO(M,4,J)
        CI(3,2,J)=CO(M+1,3,J)
```

```
      CI(4,2,J) = CO(M + 1,4,J)
**
      CI(1,3,J) = CO(M + 1,1,J)-CO(M,1,J)
      CI(1,4,J) = CO(M + 1,1,J)-CO(M,1,J)
      CI(2,3,J) = CO(M + 1,2,J)-CO(M,2,J)
      CI(2,4,J) = CO(M + 1,2,J)-CO(M,2,J)
**
      CI(3,3,J) = 0.0
      CI(3,4,J) = 0.0
      CI(4,3,J) = 0.0
      CI(4,4,J) = 0.0
 110  CONTINUE
      NUW = 3
      CALL PPSRF(CI,NUW,PT)
**
********************** draw w = constant curves on surface **************
**
      DO 120 I = 1,NUW + 1
        DO 130 J = 1,NUW
          DO 140 L = 1,3
            PT1(L) = PT(L + 2,I,J)*100.
            PT2(L) = PT(L + 2,I,J + 1)*100.
 140      CONTINUE
        CALL LINE3D(PT1,PT2,*1000)
 130    CONTINUE
 120  CONTINUE
**
```

```
********************** draw u=constant curves on surface **************
**
      DO 150 I=1,NUW
        DO 160 J=1,NUW+1
          DO 170 L=1,3
            PT1(L)=PT(L+2,I,J)*100.
            PT2(L)=PT(L+2,I+1,J)*100.
170       CONTINUE
        CALL LINE3D(PT1,PT2,*1000)
160     CONTINUE
150   CONTINUE
100   CONTINUE
      GOTO 2000
1000  WRITE(*,*) 'error in drawing surfaces, surf'
      STOP
2000  RETURN
      END
*****************************************************************************
*
*
*       SUBROUTINE PSRF
*
*       Purpose:  compute x,y,z coordinates of a point on a bicubic
*                 patch using the geometric form of the patch
*
*       CI(4,4,3)  input array of geometric coefficients defining
*                  patch
*       U          input value of parametric variable u
```

```
*       W       input value of parametric variable w
*       P(3)    output array of x,y,z coordinates of point
*
**********************************************************************
      SUBROUTINE PSRF(CI,U,W,P)
      REAL CI(4,4,3),P(3),FU(4),FW(4),A(4,3)
********** MAKE SURE U AND W ARE BETWEEN 0 AND 1 ********************
      IF (U.LT.0.0) WRITE(*,*) 'u is too small!!!!',U
      IF (U.GT.1.0) WRITE(*,*) 'u is too big!!!!',U
      IF (W.LT.0.0) WRITE(*,*) 'w is too small!!!!',W
      IF (W.GT.1.0) WRITE(*,*) 'w is too small!!!!',W
**************** EVALUATE THE BLENDING FUNCTIONS **********************
      CALL BF(U,FU)
      CALL BF(W,FW)
********************** MULTIPLY F(U)*B ********************************
      DO 100 L=1,3
      DO 200 M=1,4
        A(M,L)=(FU(1)*CI(1,M,L))+(FU(2)*CI(2,M,L))+(FU(3)*CI(3,M,L))
     $+(FU(4)*CI(4,M,L))
  200 CONTINUE
  100 CONTINUE
********************** MULTIPLY (F(U)*B)*F(W)T
*************************FAN13230
      DO 300 L=1,3
      P(L)=(A(1,L)*FW(1))+(A(2,L)*FW(2))+(A(3,L)*FW(3))+(A(4,L)*FW(4))
  300 CONTINUE
      RETURN
```

```
      END
*
*************************************************************************
*
*       SUBROUTINE PPSRF
*
*       Purpose:  compute x,y,z coordinates of rectangular array
*                 of points on a bicubic patch at specified
*                 incremental values of u and w
*
*       CI(4,4,3)   input array of geometric coefficients of patch
*       NUW         input number of increments of u and w
*       P(5,NUW+1)  output array of u,w,x,y,z
*
*************************************************************************
      SUBROUTINE PPSRF(CI,NUW,P)
      REAL CI(4,4,3), PT(3),P(5,NUW+1,NUW+1)
************** INCREMENT U AND W AND CALL SUBROUTINE PSRF ************
      DO 100 IU=1,NUW+1
      DO 200 IW=1,NUW+1
      U=(FLOAT(IU)/FLOAT(NUW))-(1./FLOAT(NUW))
      W=(FLOAT(IW)/FLOAT(NUW))-(1./FLOAT(NUW))
      CALL PSRF(CI,U,W,PT)
      P(1,IU,IW)=U
      P(2,IU,IW)=W
      P(3,IU,IW)=PT(1)
      P(4,IU,IW)=PT(2)
      P(5,IU,IW)=PT(3)
```

```
 200  CONTINUE
 100  CONTINUE
      RETURN
      END
*
*
*************************************************************************
*
*       SUBROUTINE BF
*
*       Purpose:  Calculate 4 components of blending function F at
*                 any value of parametric variable u.
*
*               U:    input value of parametric variable u
*               F(4):  output array of F
*
*
*************************************************************************
      SUBROUTINE BF(U,F)
      REAL F(4)
      F(1) = (2.*U**3)-(3.*U**2) + 1
      F(2) = (-2.*U**3) + (3.*U**2)
      F(3) = (U**3)-(2.*U**2) + U
      F(4) = (U**3)-(U**2)
      RETURN
      END
*
*
```

```
************************************************************************
*
*       SUBROUTINE BFU
*
*       Purpose:  Calculate 4 components of blending function
*                 derivative at a specified u.
*
*             U:     input value of parametric variable u
*             FU(4):  output array of FU
FAN13920
*
************************************************************************
      SUBROUTINE BFU(U,FU)
      REAL FU(4)
      FU(1)=(6.*U**2)-(6.*U)
      FU(2)=(-6.*U**2)+(6.*U)
      FU(3)=(3.*U**2)-(4.*U)+1
      FU(4)=(3.*U**2)-(2*U)
      RETURN
      END
************************************************************************
*
*       SUBROUTINE ROTVAR
*
*       Purpose:  calculate parameters used to rotate a blade
*
************************************************************************
      SUBROUTINE ROTVAR(RHUB,SC,D,NROT,ANGLE)
```

```
      PI = 3.14159
      THETA = (D*SC)/RHUB
      NROT = INT((2.*PI)/THETA)
      ANGLE = (2.*PI)/FLOAT(NROT)
      RETURN
      END
*************************************************************************
*
*       SUBROUTINE ROTMAT(ANGLE,RMAT)
*
*       Purpose:  calculate the rotation matrix required to
*                 rotate a blade
*
*************************************************************************
      SUBROUTINE ROTMAT(ANGLE,RMAT)
      REAL RMAT(3,3)
         RMAT(1,1) = 1.0
         RMAT(1,2) = 0.
         RMAT(1,3) = 0.0
**
         RMAT(2,1) = 0.0
         RMAT(2,2) = COS(ANGLE)
         RMAT(2,3) = SIN(ANGLE)
**
         RMAT(3,1) = 0.0
         RMAT(3,2) = -1.*SIN(ANGLE)
         RMAT(3,3) = COS(ANGLE)
         RETURN
```

```
      END
*************************************************************************
*
*       SUBROUTINE ROTSRF(CO,RHUB,SC,D)
*
*       Purpose:  Draw rotated blade surfaces
*
*
*************************************************************************
      SUBROUTINE ROTSRF(CO,RHUB,SC,D)
      REAL CO(9,4,3),PT(5,4,4),CI(4,4,3),PT1(3),PT2(3)
     $,RMAT(3,3),PTR1(3),PTR2(3)
      CALL ROTVAR(RHUB,SC,D,NROT,DANG)
*     DO 100 N = 1,NROT + 1
        ANGLE = (FLOAT(N)*DANG)-DANG
        CALL ROTMAT(ANGLE,RMAT)
        DO 200 M = 1,8
          IF (M.EQ.1) THEN
            M1 = 1
            M2 = 5
          ENDIF
          IF (M.EQ.2) THEN
            M1 = 5
            M2 = 9
          ENDIF
          DO 210 J = 1,3
            CI(1,1,J) = CO(1,1,J)
            CI(2,1,J) = CO(1,2,J)
```

```
            CI(1,2,J) = CO(2,1,J)
            CI(2,2,J) = CO(2,2,J)
**
            CI(3,1,J) = CO(1,3,J)
            CI(4,1,J) = CO(1,4,J)
            CI(3,2,J) = CO(2,3,J)
            CI(4,2,J) = CO(2,4,J)
**
            CI(1,3,J) = CO(2,1,J)-CO(1,1,J)
            CI(1,4,J) = CO(2,1,J)-CO(1,1,J)
            CI(2,3,J) = CO(2,2,J)-CO(1,2,J)
            CI(2,4,J) = CO(2,2,J)-CO(2,2,J)
**
            CI(3,3,J) = 0.
            CI(3,4,J) = 0.
            CI(4,3,J) = 0.
            CI(4,4,J) = 0.
 210      CONTINUE
          NUW = 3
          CALL PPSRF(CI,NUW,PT)
**
********************** draw u = constant curves on surface **************
**
      DO 220 I = 1,NUW + 1
        DO 230 J = 1,NUW
          DO 240 L = 1,3
            PT1(L) = PT(L + 2,I,J)*100.
            PT2(L) = PT(L + 2,I,J + 1)*100.
```

```
240       CONTINUE
          DO 245 L=1,3
            PTR1(L)=(PT1(1)*RMAT(1,L))+(PT1(2)*RMAT(2,L))+
     $     (PT1(3)*RMAT(3,L))
            PTR2(L)=(PT2(1)*RMAT(1,L))+(PT2(2)*RMAT(2,L))+
     $     (PT2(3)*RMAT(3,L))
245       CONTINUE
          CALL LINE3D(PTR1,PTR2,*1000)
230       CONTINUE
220   CONTINUE
**
********************** draw w=constant curves on surface **************
**
      DO 250 I=1,NUW+1
        DO 260 J=1,NUW
          DO 270 L=1,3
            PT1(L)=PT(L+2,I,J)*100.
            PT2(L)=PT(L+2,I,J+1)*100.
270       CONTINUE
          DO 275 L=1,3
            PTR1(L)=(PT1(1)*RMAT(1,L))+(PT1(2)*RMAT(2,L))+
     $     (PT1(3)*RMAT(3,L))
            PTR2(L)=(PT2(1)*RMAT(1,L))+(PT2(2)*RMAT(2,L))+
     $     (PT2(3)*RMAT(3,L))
275       CONTINUE
          CALL LINE3D(PT1,PT2,*1000)
260       CONTINUE
250   CONTINUE
```

```
200  CONTINUE
*100  CONTINUE
     GOTO 2000
1000  WRITE(*,*) 'error in drawing surfaces, surf'
     STOP
2000  RETURN
     END
*************************************************************************
*
*       SUBROUTINE ROTPT
*
*       Purpose:  rotate and draw a point
*
*
*************************************************************************
     SUBROUTINE ROTPT(CI,ANGLE,R1I)
     REAL CI(9,26,3),PT(3),PTR(3),RMAT(3,3)
     CALL ROTMAT(ANGLE,RMAT)
     DO 100 M = 1,9
       DO 110 J = 1,26,2
         DO 120 K = 1,3
           PT(K) = (CI(M,J,K))*100.
120        CONTINUE
       DO 130 K = 1,3

130     CONTINUE
       CALL PT3D(PTR,*1000)
110     CONTINUE
```

```
100  CONTINUE
     GOTO 2000
1000  WRITE(*,*) 'error in rotpt'
      STOP
2000  RETURN
      END
*************************************************************************
*
*       SUBROUTINE ROTLN
*
*       Purpose:  rotate and draw a point
*
*
*************************************************************************
      SUBROUTINE ROTLN(CI,ANGLE,R1I,NS,CIR)
      REAL
      CALL ROTMAT(ANGLE,RMAT)
      DO 100 M = 1,NS
        DO 110 J = 1,26
          DO 120 K = 1,3
            PT(K) = (CI(M,J,K))*100.
120       CONTINUE
        DO 130 K = 1,3

          CIR(M,J,K) = PTR(K)
130     CONTINUE
110     CONTINUE
100  CONTINUE
```

```
      DO 140 M = 1,NS
        DO 150 J = 2,24,2
          DO 160 K = 1,3
            PT1(K) = CIR(M,J,K)
            PT2(K) = CIR(M,J + 2,K)
160   CONTINUE
            CALL LINE3D(PT1,PT2,*1000)
150   CONTINUE
140   CONTINUE
      DO 170 M = 1,NS-1
      DO 180 K = 1,3
       PT1(K) = CIR(M,1,K)
       PT2(K) = CIR(M + 1,1,K)
180   CONTINUE
      CALL LINE3D(PT1,PT2,*1000)
170   CONTINUE
      DO 175 M = 1,NS-1
      DO 185 K = 1,3
       PT1(K) = CIR(M,26,K)
       PT2(K) = CIR(M + 1,26,K)
185   CONTINUE
      CALL LINE3D(PT1,PT2,*1000)
175   CONTINUE
      GOTO 2000
1000  WRITE(*,*) 'error in rotln'
      STOP
2000  RETURN
```

```
      END
**
**
**
*************************************************************************
*
*     Subroutine:  OUT1
*
*     Purpose:     record results of computations in output file
*
*************************************************************************
      SUBROUTINE OUT1(MF,DPT,VX1,BR1MD,BR2MD,WP,WA,WS,EF2,SPSPD,SPDIA)
      REAL MF,DPT,VX1,BR1MD,BR2MD,WP,WA,WS,EF2,SPSPD,SPDIA
        WRITE (3,*) ' '
        WRITE (3,*) ' '
        WRITE (3,*) 'MASS FLOWRATE=', MF, 'KG/S'
        WRITE (3,*) 'TOTAL PRESSURE CHANGE=', DPT, 'kPa'
        WRITE (3,*) 'AXIAL VELOCITY=',VX1,'M/S'
        WRITE (3,*) ' '
        WRITE (3,*) 'SPECIFIC SPEED=',SPSPD
        WRITE(3,*)  'SPECIFIC DIAMETER=',SPDIA
        WRITE (3,*) ' '
        WRITE(3,*) ' '
        WRITE(3,*)
        WRITE(3,*) ' '
        WRITE(3,*) 'RESULTS OF MEANLINE DESIGN:'
        WRITE(3,*) ' '
        WRITE (3,*) 'MEAN INLET ROTOR ANGLE=', BR1MD, 'DEGREES'
```

```
      WRITE (3,*) 'MEAN OUTLET ROTOR ANGLE=', BR2MD, 'DEGREES'
      WRITE (3,*) ' '
      WRITE (3,*) 'MEANLINE LOSSES:'
      WRITE (3,*) 'Profile loss coefficient WP=  ', WP
      WRITE (3,*) 'Annulus loss coefficient WA=  ', WA
      WRITE (3,*) 'Secondary loss coefficient WS=', WS
      WRITE (3,*) ' '
      WRITE (3,*) 'EFFICIENCY=',EF2
      WRITE (3,*) ' '
      WRITE(3,*)
      WRITE (3,*) ' '
      WRITE (3,*) ' '
      RETURN
      END
**
**
**
*************************************************************************
*
*    Subroutine:  OUT2
*
*    Purpose:     record results of computations in output file
*
*************************************************************************
    SUBROUTINE OUT2(R,BR1D,BR2D,NSEC)
    REAL BR1D(10),BR2D(10),R(10)
    INTEGER NSEC
    WRITE (3,*) 'INLET AND OUTLET BLADE ANGLES, AS A FUNCTION
```

```
     $OF RADIUS'
      WRITE (3,*) ' '
      WRITE (3,*) '  RADIUS',' B1',' B2'
      WRITE (3,*) ' '
      DO 10 I = 1,2*NSEC + 1
      WRITE (3,*) R(I), BR1D(I),BR2D(I)
   10  CONTINUE
      RETURN
      END
**
**
**
*************************************************************************
*
*    Subroutine:  OUT3
*
*    Purpose:     record results of computations in output file
*
*************************************************************************
      SUBROUTINE OUT3(NS,R,RARC,IBR1D,IBR1M,IBR1S,IBR2D,IBR2M,IBR2S,
     $ST,ED,CT)
      REAL R(NS),RARC(NS),IBR1D(NS),IBR1M(NS),IBR1S(NS),IBR2D(NS)
     $,IBR2M(NS),IBR2S(NS),ST(NS,3),ED(NS,3),CT(NS,3)
      INTEGER NSEC
      WRITE (3,*) ' '
      WRITE (3,*) ' '
      WRITE(3,*) '*************************************************'
      WRITE (3,*) ' '
```

```
      WRITE (3,*) 'GEOMETRY REQUIRED BY CADAM'
      WRITE (3,*) ' '
      WRITE (3,3000)
 3000 FORMAT (5X,'R   ',6X,'RARC',14X,'B1',23X,'B2')
      WRITE (3,3100)
 3100 FORMAT (26X,'DEG',3X,'MIN',3X,'SEC',10X,'DEG',3X,'MIN',3X,
     $'SEC')
      WRITE (3,*) ' '
      DO 100 I = 1,NS
      WRITE (3,200) R(I),RARC(I),IBR1D(I),IBR1M(I),
     $IBR1S(I),IBR2D(I),IBR2M(I),IBR2S(I)
  200 FORMAT (2X,F8.4,2X,F8.4,5X,I3,4X,I3,3X,I3,9X,I3,4X,I3,3X,I3)
  100 CONTINUE
      WRITE (3,*) ' '
      WRITE (3,*) ' '
      WRITE (3,300)
  300 FORMAT (5X,'SX',10X,'SY',8X,'EX',9X,'EY',9X,'CX',9X,'CY')
      DO 500 I = 1,NS
      WRITE (3,400) ST(I,1),ST(I,2),ED(I,1),ED(I,2),CT(I,1),CT(I,2)
  400 FORMAT (2X,F8.6,3X,F8.6,3X,F8.6,3X,F8.6,3X,F8.6,3X,F8.6)
  500 CONTINUE
      RETURN
      END
**
**
**
*************************************************************************
*
```

```
*    Subroutine:  OUT4
*
*    Purpose:     record results of computations in output file
*
**********************************************************************
      SUBROUTINE OUT4(PTOT,ETOT,CTOT,EF3)
      REAL PTOT,ETOT,CTOT,EF3
        WRITE (3,*) ' '
        WRITE(3,*) ' '
        WRITE(3,*)
        WRITE(3,*) ' '
        WRITE(3,*) 'RESULTS OF 3-DIMENSIONAL DESIGN:'
        WRITE(3,*) ' '
        WRITE (3,*) '3-DIMENSIONAL LOSSES:'
        WRITE (3,*) 'Profile loss coefficient WP=  ', PTOT
        WRITE (3,*) 'Endwall loss coefficient WA=  ', ETOT
        WRITE (3,*) 'Clearance loss coefficient WS=', CTOT
        WRITE (3,*) ' '
        WRITE (3,*) 'EFFICIENCY=',EF3
        WRITE (3,*) ' '
        WRITE(3,*)
        WRITE (3,*) ' '
        WRITE (3,*) ' '
        RETURN
        END
```

Appendix D.

FAN3D: Sample Output

MASS FLOWRATE= 75.0000000 KG/S

TOTAL PRESSURE CHANGE= 2.50000000 kPa

AXIAL VELOCITY= 115.395844 M/S

SPECIFIC SPEED= 4.57717419

SPECIFIC DIAMETER= 0.847624540

****

RESULTS OF MEANLINE DESIGN:

MEAN INLET ROTOR ANGLE= 31.6573486 DEGREES

MEAN OUTLET ROTOR ANGLE= 1.74544048 DEGREES

MEANLINE LOSSES:

Profile loss coefficient WP= 0.946115702E-02

Annulus loss coefficient WA= 0.185009874E-01

Secondary loss coefficient WS= 0.180073939E-01

EFFICIENCY= 0.780645967

****

INLET & OUTLET BLADE ANGLES, AS A FUNCTION OF RADIUS

RADIUS	B1	B2
0.250000000	22.3431854	-32.1087036
0.293151140	26.1997223	-20.7630920
0.330719292	29.5572968	-12.1243296
0.364435017	32.5705566	-5.27725220
0.395285189	35.3277283	0.303040981
0.423896134	37.8847656	4.94648457
0.450694501	40.2798157	8.87353134

****

GEOMETRY REQUIRED BY CADAM

R	RARC	B1			B2		
		DEG	MIN	SEC	DEG	MIN	SEC
0.2500	0.1097	22	20	35	-32	-6	-31
0.2932	0.1256	26	11	59	-20	-45	-47
0.3307	0.1422	29	33	26	-12	-7	-27
0.3644	0.1587	32	34	14	-5	-16	-38
0.3953	0.1745	35	19	39	0	18	10
0.4239	0.1894	37	53	5	4	56	47

0.4507 0.2031 40 16 47 8 52 24

SX	SY	EX	EY	CX	CY
0.000000	0.000000	0.100000	-.008543	0.041698	-.101453
0.000000	0.000000	0.100000	0.004748	0.055465	-.112720
0.000000	0.000000	0.100000	0.015332	0.070137	-.123678
0.000000	0.000000	0.100000	0.024279	0.085408	-.133700
0.000000	0.000000	0.100000	0.032136	0.100923	-.142393
0.000000	0.000000	0.100000	0.039221	0.116335	-.149521
0.000000	0.000000	0.100000	0.045734	0.131335	-.154976

****

RESULTS OF 3-DIMENSIONAL DESIGN:

3-DIMENSIONAL LOSSES:

Profile loss coefficient WP = 0.509635359E-02

Endwall loss coefficient WA = 0.339757022E-02

Clearance loss coefficient WS = 0.499926880E-02

EFFICIENCY = 0.762215930

****

Bibliography

1. Abbott, Ira H., and Von Doenhoff, Albert E., *Theory of Wing Sections*, Dover Publications, 1959.

2. Balje, O.E., *Turbomachines--A Guide to Design, Selection, and Theory* , John Wiley and Sons, 1981.

3. CADAM Inc., *Computer-Graphics Augmented Design and Manufacturing System--User Training Manual SH20-2035-3*, IBM, 1982.

4. CADAM Inc., *CADAM System--Geometry Interface Module Installation and Programmer's Guide SH20-2099-6*, IBM, 1983.

5. CADAM Inc.,*CADAM System--Geometry Interface Installation Guide SH20-6227-0*, IBM, 1985.

6. Clancy, L.J., *Aerodynamics*, John Wiley and Sons, 1975.

7. Csanady, G.T., *Theory of Turbomachines*, McGraw-Hill , 1964.

8. Dixon, S.L., *Fluid Mechanics, Thermodynamics of Turbomachinery,* , 3rd ed., Pergamon Press, 1978.

9. Eck, Bruno, *Fans- Design and Operation of Centrifugal, Axial-Flow and Cross-Flow Fans*, translated and edited by Ram S. Azad and David R. Scoot, Pergamon Press, 1973.

10. Fielding, Leslie, *Handheld Calculator Programs for Rotating Equipment Design*, McGraw-Hill, 1983.

11. Herrig, L. Joseph, James C. Emery, and John R. Erwin, *Systematic Two-Dimensional Cascade Tests of NACA 65-Series Compressor Blades at Low Speeds*, NACA TN 3916, 1957.

12. Horlock, J.H., *Axial-Flow Compressors, Fluid Mechanics and Thermodynamics*, Robert E. Krieger Publishing Co., 1973.

13. Johnsen, Irving A., and Robert O. Bullock, "Aerodynamic Design of Axial-Flow Compressor", Revised Edition, NASA Publication SP-36, 1965.

14. Keller, Curt, *Theory and Performance of Axial-Flow Fans* ,McGraw-Hill, 1937.

15. Kuethe, A.M., and J. D. Schetzer, *Foundations of Aerodynamics* , John Wiley and Sons, 2d ed., 1959.

16. Krouse, John K., *What Every Engineer Should Know about Computer-Aided Design and Computer-Aided Manufacturing*, Marcel Dekker, Inc., 1982.

17. Logan, Earl Jr., *Turbomachines--Basic Theory and Applications* , Marcel Dekker, Inc., 1981.

18. Macchi, Ennio, "Design Limits, Basic Parameter Selection and Optimization Methods in Turbomachinery Design,"*Thermodynamics and Fluid Mechanics of Turbomachinery, Volume II*, Martinus Nijhoff Publishers, 1985, pp.805-828.

19. Madison, Richard D., Editor, *Fan Engineering--An Engineer's Handbook*, Buffalo Forge Co., 5th ed., 1948.

20. Mortenson, Michael, *Geometric Modelling*, John Wiley and Sons, 1985.

21. Moses, H. L., *Lecture Notes on Turbomachinery*, Mechanical Engineering Department, VPI&SU.

22. Nystrom, Lynn, "The Automation of Design", College of Engineering, Virginia Polytechnic Institute and State University, unpublished.

23. Osborne, William C., *Fans--Volume I*, Pergamon Press, 1966.

24. Ozell, Benoit, and Ricardo Camerero, "Between the Blades-- Geometric and Computational Modeling of Turbomachinery", *Computers in Mechanical Engineering*, January 1986, pp28-35.

25. Pollack, Robert, "Selecting Fans and Blowers", *Fluid Movers--Pumps, Compresssors, Fans, and Blowers*, Jay Matley, Editor, McGraw-Hill, 1979.

26. Serovy, George K., "Organization of Design Systems for Turbomachinery", *Thermodynamics and Fluid Mechanics of Turbomachinery, Volume II*, Martinus Nijhoff Publishers, 1985, pp.785-803.

27. Shepherd, D.G., *Principles of Turbomachinery*, Macmillan Co., 1956.

28. Stow, Peter, "Turbomachinery Design Using Advanced Calculation Methods" , *Thermodynamics and Fluid Mechanics of Turbomachinery, Volume II*, Martinus Nijhoff Publishers, 1985, pp.923-964.

29. Stribling, C.B., *Basic Aerodynamics*, Butterworths Scientific Publications, 1984.

30. Turton, R.K., *Principles of Turbomachinery*, E. & F.N. Spon, 1984.

31. Vavra, Michael H., *Aerodynamics and Flow in Turbomachines*, John Wiley and Sons, 1960.

32. Wallis, R. Allan, *Axial Flow Fans and Ducts*, John Wiley and Sons, 1983.

The vita has been removed from the scanned document