Aerodynamic Shape Optimization of Fan Blades

by

Timothy P. Rogalsky

B.R.S. (Concord College) 1991
B.Sc. (University of Manitoba) 1996

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Master of Science

in

Applied Mathematics

at the
UNIVERSITY OF MANITOBA

Committee in charge:

Dr. Serpil Kocabiyik, Chair
Dr. Robert Derksen

1998

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

AERODYNAMIC SHAPE OPTIMIZATION OF FAN BLADES

BY

TIMOTHY P. ROGALSKY

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Abstract

Aerodynamic Shape Optimization of Fan Blades

by

Timothy P. Rogalsky

Master of Science in Applied Mathematics

University of Manitoba

Dr. Serpil Kocabiyik, Chair

The purpose of this work is to develop and evaluate an inverse optimization algorithm which designs two-dimensional fan blade shapes. Given a prescribed pressure distribution and inlet and outlet flow angles, this design optimization technique finds the optimal fan blade shape, stagger angle, and pitch/chord ratio. The algorithm is coded into a completely self-contained C++ program. Its three main components are: a surface vorticity panel method flow solver, a Bezier curve surface definition routine, and an optimization method. Three different optimizers are tested and compared. A relatively new genetic algorithm, Differential Evolution, is determined to be the most effective. To demonstrate the abilities of the aerodynamic shape optimization algorithm, several fan blades are designed to exhibit a Liebeck pressure distribution. For each design, the optimal fan blade spacing is also found, verifying theoretically a claim that until now has been supported experimentally and with simple modelling.

Acknowledgements

I would like to thank my advisors, Serpil Kocabiyik and Robert Derksen, for suggesting the problem and for their patience and advice as I completed it. Thank-you, Bryant Moodie, for agreeing to examine this thesis. It is an honour, and your comments are much appreciated. Thanks to Harley Cohen, without whose financial support I would not have entered into post-graduate studies. Jason Bender's assistance in locating some of the references is much appreciated. Finally, to Deb, Nathan, Lois, and Mark, thank-you for the support and freedom you have given me throughout this project.

Contents

Nomenclature

Acronyms

CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
DE	Differential Evolution
DNA	Deoxyribonucleic Acid
DS	Downhill Simplex
NFE's	Number of Function Evaluations
PC	Personal Computer (Intel-based)
SA	Simulated Annealing

Roman Symbols

b_i	Bezier control variables necessary to define an airfoil shape
$B_{i,n}$	blending function
$C(n,i)$	binomial coefficient
C_L	surface lift coefficient
c_m	y-values of camber curve
C_P	surface pressure coefficient computed with respect to inlet velocity W_1
$C_{P\alpha}$	surface pressure coefficient computed with respect to constant velocity W_α
CR	crossover constant in differential evolution
D	dimension of a differential evolution optimization problem
E	energy in a simulated annealing optimization problem
$\mathbf{e}_i$	unit vector
F	differential weight for mutation in differential evolution
f	objective function to be optimized
K	coupling coefficient matrix
$K(s_m,s_n)$	coupling coefficient

l	chord (cross-sectional length of a fan blade)
M	number of panels
m	number of iterations for an annealing schedule
N	number of control variables in optimization problem
NP	differential evolution population size
$\mathbf{P}$	Bezier parametric curve function
$\mathbf{P}_i$	Bezier curve control points
r_{mn}	straight line distance between points s_m and s_n
S_m	pivotal point at center of panel number m
s_m	panel endpoint on body surface
T	annealing schedule temperature
T_0	initial annealing schedule temperature
t	pitch of a fan (distance between consecutive fan blades)
t_m	y-values of thickness curve
te	trailing edge
u	parameter of the Bezier curves
U_∞	x-component of W_∞
$v(x)$	velocity of the flow at position x along the chord
v_s	velocity of the flow around a body at point s, just outside the boundary layer
V_∞	y-component of W_∞
W_1	inlet velocity
W_2	outlet velocity
W_∞	uniform stream
$\mathbf{x}$	control variables used in optimization problem
x_m	x-values of thickness and camber curves
(X_m, Y_m)	panel endpoints
(x_m, y_m)	pivotal point at center of panel number m

Greek Symbols

β_1	angle of inlet velocity
β_2	angle of outlet velocity
β_{2s}	angle of outlet velocity
β_{2t}	angle of outlet velocity
β_∞	angle of flow of the uniform stream
ε	temperature reduction in an annealing schedule
$\Delta\Gamma_m$	circulation around the interior of a profile
Δs_m	length of panel m
ΔT_m	thickness of the body at pivotal point S_m
Γ	blade bound circulation
Γ_u	x-component of the unit bound circulation

Γ_v	y-component of the unit bound circulation
$\gamma(s)$	vorticity strength per unit length at point s
λ	stagger angle of the blades of a fan
μ	characteristic length scale of a solution space
ρ_m	slope of the boundary at point s_m with respect to the x-axis
σ_i	pressure distribution of test fan blade
τ_i	target pressure distribution

List of Figures

List of Tables

Chapter 1

Introduction

Optimization and design are of interest in any industry. Aerodynamic shape optimization involves designing the best shapes of bodies that move through fluids, such as fan blades, ship hulls, or aircraft wings. For example one might require a fan blade that will maximize the air flow through a fan, a hull shape that will minimize the drag of a ship, or an airfoil that will maximize the lift of an airplane. To generate these optimal shapes, the assistance of computers is nearly always required, and programs may take many hours to run.

While commercial fans have been in use for many years, research continues into efficient methods for optimizing the shape of the fan blades based on a given set of design criteria. Most computational methods for predicting flow through fans are analysis methods. These predict the pressure distribution on the surface of a fan blade of specified geometry, including the effects of the cascade of blades surrounding it. Relatively few methods, however, address the aerodynamic designer's responsibility of designing more efficient fan blade shapes. It is the latter task, known as design optimization, that is the

focus of this work.

Although few fan blade design methods exist, the approaches taken to solve the problem of design optimization vary widely. Most of these approaches can be categorized into two types: direct and indirect design methods.

Direct design techniques solve for the desired airfoil shape by minimizing a given aerodynamic objective function. For example, the drag to lift ratio might be minimized, which would be equivalent to maximizing the lift to drag ratio. The optimizer perturbs the airfoil until it finds the shape which produces the lowest possible objective function value. In these methods, the designer may directly constrain the airfoil to meet necessary geometric constraints. The final converged solution is then obtained by minimizing the objective function while still satisfying all the constraints.

Direct methods allow for natural problem formulation and offer the airfoil designer great flexibility in satisfying constraints from various sources. In the early days of aerodynamic engineering, candidate geometries were evaluated in wind tunnels, then physically modified and retested by the designer. Now, sophisticated computational fluid dynamics (CFD) techniques exist to calculate the flow accurately. These techniques require a solution of the full Navier-Stokes equations, however, and are computationally unwieldy. Thus direct optimization methods may require substantial computational time to obtain a converged solution, making them both expensive and inconvenient for use as an everyday design procedure. In addition, if the design space is very complex, the solution obtained may represent only a local minimum and not the globally optimal configuration.

The indirect, or inverse, method uses an optimization function to design a shape

that will exhibit a prescribed aerodynamic distribution, usually the surface pressure distribution. The quality of the optimized shape depends on how well the distribution is defined. The design objective in the case of the inverse problem is to minimize the deviation between the target distribution and the distribution corresponding to the current geometry.

The principle advantages of inverse design techniques are speed and reliability. Less sophisticated flow solvers are required. Thus they are much faster than the direct design methods, and can even allow interactive design at workstations. Further, since the objective function is a measure of the deviation from a specified distribution, the optimal objective function value is known to be zero. The designer can thus easily evaluate the performance of the designed blade to determine whether or not a global minimum has been attained.

In this thesis, an inverse design optimization software package is developed to design the cross-sectional shape of a fan blade. The input parameters include a desired pressure distribution over the blades, an initial fan blade shape, and fan design variables such as inlet and outlet angles of the flow. Three essential elements characterize the method: first, a module for implementing the aerodynamic calculations; second, a procedure to describe and change the fan blade geometry through design variables; third, an optimization method. An objective function first calculates the geometry, then the corresponding pressure distribution, and finally the deviation of this flow from the target pressure distribution. Optimization is then done by searching among candidate solutions for those that minimize the objective function.

The aerodynamic calculations are performed using a panel method flow solver.

Fluid dynamics techniques are used to closely approximate the velocity of the flow at a distribution of points on the airfoil. The corresponding pressure coefficients at these points are then computed, giving a pressure distribution around the fan blade.

Flow calculations on an enormous number of airfoils must be performed in design optimization. Modern CFD based analysis, in spite of the resources and advancement, cannot hope to successfully satisfy this requirement in a reasonable time. In inverse design, however, the aerodynamic analysis is more important as a filter to allow candidate solutions than to accurately estimate the coefficients. Once the optimum has been identified, then a final flow simulation can be attempted using a sophisticated CFD package. In this respect, the panel method provides an admirable low cost alternative for exploring the solution space.

The fan blade geometry is calculated using Bezier curves, a method that is relatively new to aerodynamics. They were first used by the French firm Regie Renault to design the outer panels of automobiles. In addition to being easy to calculate and perturb, they have the advantage of requiring relatively few variables to describe a realistic airfoil shape.

In general, an optimization routine considers an objective function which depends on a fixed number of variables. It searches the design space for the variables which will minimize that function. For this project the objective function measures the error between the pressure distribution required and the pressure around the fan blade shape defined by the variables. In order to search for the optimal (or near-optimal) set of variables, many function calls are made, requiring many flow calculations and high CPU time.

The aerodynamic shape optimization code developed in this thesis will be used to produce a number of results. It will facilitate comparison of different optimization algorithms as applied to aerodynamic design. The effectiveness of Bezier parametrization as an aerodynamic design tool will be illustrated. Finally, the code will be used to demonstrate the ability of the scheme presented to design optimally shaped fan blades. In this design process, an important discovery is made about the optimal spacing of blades within a fan.

Chapter 2

Literature Review

Aerodynamic optimization programs typically require three separate modules: a shape definition routine, a flow solver, and an optimization algorithm. These represent three distinct and independent research fields. Since aerodynamic design draws from and combines these fields of inquiry, the literature surrounding each individual topic is reviewed. The manner in which aerodynamic optimization has combined the research will also be examined.

2.1 Flow Solver

Aerodynamic calculations around a moving body immersed in fluid are performed numerically by a flow solver. Panel methods discretize the body shape into straight line segments, called panels, in order to approximate the velocity of the potential flow close to the body profile. Other aerodynamic quanitities, such as pressure and the lift and drag forces, can then be computed from the velocity field.

Modern, sophisticated CFD methods exist which can accurately compute the flow field. In spite of resources and advancement, however, they are still too computationally intensive to be feasible in an optimization scheme. Panel methods, while less accurate than others, do give reasonable estimates of the flow, and are still being usefully applied in other situations (Pfeiffer, 1990). More importantly, they are able to simulate the flow quickly enough to be effective even in schemes for which many iterations are required.

The origins of the panel method can be found in classical mathematics. Kellog (1929) wrote a comprehensive book dealing with potential theory by the use of integral equations. Incompressible inviscid flow is modelled by Laplace's equation for the velocity potential. Integrating an elementary solution over the body surface, Kellog developed an integral equation that represented the flow past a body immersed in a uniform stream. Panel methods approximate these integrals by discretizing the curves into panels, and then integrating numerically.

Two techniques use panels to represent the surface: the source panel method, and the surface vorticity method. Source panel methods have been widely used with great success since about 1953. Representative early work was done by Hess (1962, 1967) and Smith (Hess and Smith, 1966; see also Smith, 1990 for an overview). One limitation of the technique is that it cannot be applied to lifting bodies. The model is not based on the physical reality of fluid flow around bodies, but is effective mathematically. Each panel is considered to be a point source, and contributes to the velocity of the potential flow on every other panel. Kellog's integral equation can then be written for each pair of panels, using the Neumann boundary condition. That is, on each panel, the velocity normal to

the surface is zero.

In contrast with the source panel method, surface vorticity modelling has a direct physical interpretation, and is able to correctly model the flow around lifting bodies. Martensen (1959) laid out this powerful new computational technique, and extended his boundary integral theory to deal with turbomachine cascades. This procedure considers the vorticity in the viscous boundary layer adjacent to the body's surface. Martensen's integral relates the vortex strength at any given point to the vortex strengths at all other points on the surface. The Dirichlet boundary condition of zero parallel surface velocity is then imposed to solve for the vorticity on each panel. The velocity of the potential flow on the boundary can then be directly calculated, since it is a function of vorticity alone.

Martensen's method for airfoils was first successfully implemented on a digital computer by Jacob & Riegels (1963). An analysis of an airfoil with 36 elements took 15 minutes to execute, a remarkable feat for that time. The method was still not completely reliable, but Wilkinson (1967) was able to identify and resolve many modelling and computational obstacles. Wilkinson (1969) also extended his work to mixed-flow cascades.

An excellent summary of vorticity methods is given by Sarpkaya (1989). He reviews the theoretical foundations, the development of the method, and the practical applications, and includes many references to the literature.

2.2 Shape Definition

The shape definition module must deliver an efficient technique to describe and perturb the geometry of the airfoil. Several methods have been used, including linear

combination of basis shapes (Vanderplaats, 1975), basis functions (Hicks and Henne, 1977), Legendre polynomials (Coiro & Nicolosi, 1995), and Bezier polynomials (Venkataraman, 1995a). The first three techniques have significant disadvantages when used in aerodynamic optimization (Burgreen et al., 1992; Venkataraman, 1995b). Some of these are wild oscillations, large number of parameters, and inability to incorporate multidisciplinary design. The use of Bezier curves, on the other hand, is proving to be quite effective.

P. Bezier, of the French firm Regie Renault, pioneered the use of computer modeling of surfaces in automobile design. His UNISURF system, initiated in 1962 and used by designers since 1972, has been applied to define the outer panels of several cars marketed by Renault (Bezier, 1972, 1974). The foundations of Bezier curves, however, go back much further. In 1926, S. Bernstein presented a constructive proof of the Weierstrass approximation theorem (Davis, 1963) using functions that have become known as Bernstein polynomials. He showed how to construct a Bernstein polynomial that will approximate uniformly any continuous function over any closed interval. Bezier curves are very similar in form to those used by Bernstein, and are sometimes referred to as Bezier-Bernstein polynomials.

Among other advantages, Bezier curves are easily used to define airfoil shapes. An nth order Bezier curve can be defined using $n+1$ control points, which are then the control variables used to define the shape. In one of the first examples, Birckelbaw (Birckelbaw, 1989) used two 44th order Bezier curves to define the cross-sectional shape of an airfoil. The 90 control points for the curves were used as the optimization parameters. In comparison with other geometry-defining methods, convergence of the solution was much better. Also, in one test case, other techniques generated physically unrealistic airfoil surfaces, whereas

Bezier curves ensured a smooth and continuous shape throughout the design process.

Burgreen et al. (1992) were able to show that Bezier curves could be used to increase the efficiency of an aerodynamic shape optimization process. They represented the surface of an internal-external nozzle by Bezier polynomials instead of grid points. These curves can be defined using relatively few points, allowing the number of design variables to be decreased from 47 to six. Since the optimization routine then searches a smaller dimensional space, the CPU time was reduced by a factor of almost four. The representation of the nozzle surface by Bezier curves also demonstrated their ability to accurately model aerodynamic shapes other than airfoils, a significant advantage over other techniques.

The number of parameters required to represent airfoils has been reduced since the paper by Birckelbaw. Venkataraman (1995a) used four cubic Bezier curves, two for the top surface and two for the bottom, to define an airfoil cross-section. To form a closed shape, the endpoints of adjoining curves must be constrained to be coincident. Ignoring other constraints, Birckelbaw's method required 88 control points, whereas the new method needed only twelve. After imposing other reasonable constraints (such as first order continuity of the airfoil shape) on these two-dimensional control points, Venkataraman was able to reduce the number of design variables to fourteen.

Venkataraman (1996a) discovered, however, that the resolution of the leading edge region was inadequate. So he enhanced his Bezier parametrization scheme, using fourth order polynomials for the two leading edge curves. In doing so, an inflection point on the bottom surface was also accommodated. The number of design variables grew to 19,

slowing down the optimization process. However, this diminished efficiency was offset by the improvement in the flow around the resultant shapes.

2.3 Optimization Algorithms

The goal of an optimization problem can be formulated as follows: find the combination of parameters which minimize a given quantity, possibly subject to some constraints on the allowed parameter ranges. The quantity to be minimized is termed the objective function, $f(\mathbf{x})$; the vector $\mathbf{x}$ contains n parameters, called control or decision variables, which may be changed in the quest for the optimum; the restrictions on allowed parameter values are known as constraints; the allowable space which is to be searched is called the control space.

For the design optimization problem discussed here, multidimensional optimization algorithms are used that do not require calculation of the gradient of the objective function. The reason for the exclusion of gradient-based techniques is the random nature of the penalty function used by the objective function in this work. A penalty function assigns a large random number to the objective function whenever any constraints are violated. The necessity for and nature of this technique is discussed more fully below in section 3.4.3.

Three algorithms were tested on this design optimization problem. The downhill simplex method attempts to crawl slowly downhill towards the minimum function value. Simulated annealing employs a random search of the design space and will sometimes accept uphill steps in its attempt to find the global minimum. Differential evolution simulates natural selection, as theorized by Darwin, in its search for the best set of variables.

2.3.1 Downhill Simplex Method

A simplex is simply an N-dimensional figure consisting of $N + 1$ vertices and all the polygonal faces between them. Spendley et al. (1962) introduced the use of the simplex as a tracking device to find optimum operating conditions. The simplex is modified in such a way as to close in on the optimum. New simplices are formed by reflecting one point and keeping the rest stationary. The technique is a rudimentary steep ascent method which requires only trivial arithmetic, and can actually be done by hand. It compared well with a variety of other techniques common at the time.

Nelder and Mead (1965) adapted this method to the problem of minimizing a mathematical function in several variables. An initial simplex is formed of $N + 1$ vectors. The vector maximizing the objective function is then modified to form a new simplex. This vertex is now modified not just by reflection but also by contraction or expansion, allowing the simplex to adapt itself to the local topography by elongation down inclined planes, change of direction when a valley is encountered and contraction around a minimum. The technique accepts only downhill steps, and will converge to the lowest local minimum bracketed by the initial simplex.

The new method was compared to other non-gradient-based techniques, and, in some cases, was found to outperform them. It should be emphasized that the downhill simplex method is not designed to search for the global minimum of the objective function.

2.3.2 Simulated Annealing

As its name implies, the simulated annealing method exploits an analogy with the way in which metals cool and anneal. When metals are cooled slowly enough, nature is able to find the minimum energy crystalline structure by redistribution of the atoms as they lose mobility. The algorithm to find this minimum temperature can be used in the search for a minimum in a more general system.

The optimization algorithm is based upon that of Metropolis et al. (1958), which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The major advantage of the Metropolis algorithm over other methods is the ability to avoid becoming trapped at local minima by accepting some uphill steps. The connection between this algorithm and mathematical minimization was first noted by Pincus (1970), but it was Kirkpatrick et al. (1983) who proposed that it form the basis of an optimization technique for combinatorial problems.

The objective function in mathematical optimization is analogous to energy in a physical system, and the global minimum to the minimum energy state. An annealing schedule describes the temperature parameter, T, and gives rules for lowering it as the search progresses. Then, using the Metropolis algorithm, the control space is randomly examined among local minima of depth less than about T. As T is lowered, the number of such minima qualifying for frequent visits is gradually reduced.

The combinatorial algorithm has effectively solved the travelling salesman problem, and has been used successfully to design complex integrated circuits. This algorithm was extended to the case of continuous multidimensional problems by Vanderbilt and Louie

(1984). The simulated annealing approach was found to rank fairly high compared to other known algorithms. Especially attractive is its ability to find a global minimum in the presence of local minima. If the temperature is decreased too quickly, however, it can also become trapped in a local minimum. Also, the best annealing schedule is problem dependent, and many experiments are typically required before simulated annealing will be effective at a new optimization problem.

2.3.3 Differential Evolution

Differential evolution is a member of a broader class of optimization algorithms called genetic algorithms. These attempt to simulate the theory of natural evolution. In natural evolution each species searches for beneficial adaptations in an ever-changing environment. As species evolve, these new attributes are encoded in the chromosomes of individual members. This information does change by random mutation, but the real driving force behind evolutionary development is the combination and exchange of chromosomal material during breeding.

Although sporadic attempts to incorporate these principles in optimization routines have been made since the early 1960's (see a review in Chapter 4 of Goldberg, 1989), genetic algorithms were first established on a sound theoretical basis by Holland (1975). The two key axioms underlying this innovative work were that complicated nonbiological structures could be described by simple bit strings and that these structures could be improved by the application of simple transformations to these strings. One significant difference from the techniques discussed above is that genetic algorithms search from one population of solutions to another, rather than from individual to individual.

While genetic algorithms were designed to solve discrete or integer optimization problems, evolutionary strategies were applied first to continuous parameter optimization problems associated with laboratory experiments. Evolutionary strategies were introduced in the 1960's by Rechenberg (1973) working in Berlin and further developed by Schwefel (1975a). The first numerical simulations were performed by Hartmann (1974), and the first attempts at using evolutionary strategies to solve discrete optimization were made by Schwefel (1975b). Not only are evolutionary strategies significantly faster at numerical optimization than traditional genetic algorithms, they are also much more likely to find a function's true global optimum.

Differential evolution grew out of Ken Price's attempts to solve the Chebychev polynomial fitting problem that had been posed to him by Rainer Storn (Storn & Price, 1995). It is a simple yet powerful population-based, stochastic function minimizer. The crucial idea behind it is a scheme for generating trial parameter vectors. Basically, differential evolution adds the weighted difference between two population vectors to perturb a third vector. The algorithm finished third at the First International Contest on Evolutionary Computation which was held in Nagoya in May 1996, but is more universally applicable than the first two finishers. Differential evolution has not been patented in the hopes that it will be further developed by scientists around the world. The source code is freely available at http://www.icsi.berkeley.edu/~storn/code.html.

2.3.4 Comparison

Relatively few research papers have assessed the performance of optimization algorithms for aerodynamic problems. Obayashi and Tsukahara (1996) endeavoured to justify

the use of genetic algorithms in aerodynamic design, despite the large number of function evaluations they require. Using a simplified design problem of maximizing the lift of a low-speed airfoil, they compared the performance of three optimization strategies: a gradient-based method, simulated annealing, and a genetic algorithm. Although the genetic algorithm was found to be the most time consuming method, it consistently attained the global minimum. The gradient-based and simulated annealing techniques were extremely susceptible to being trapped by a local minimum. They argued that, for these latter approaches to be as effective as genetic algorithms, many local minima would have to be examined. That is, for any given problem, many optimmizations would have to be performed, each starting at different initial shapes. In this sense, genetic algorithms are more efficient, requiring fewer function evaluations to achieve the same results.

2.4 Aerodynamic Optimization

Aerodynamic optimization has been used since the 1960's to design various airfoils. Direct design techniques solve for the desired airfoil shape by optimizing a given aerodynamic objective function, such as lift or drag. The optimizer perturbs the airfoil until it finds the shape which produces the best possible value. Inverse design methods design shapes to exhibit a prescribed aerodynamic distribution, usually the surface pressure distribution. The design objective in the case of the inverse problem is the deviation between the distributions of the target and of the current geometry.

For the inverse design approach, the quality of the optimized shape will obviously depend on how well the distribution is defined. An initial step toward designing optimal

pressure distributions was given by Stratford (1959a), who introduced a method for predicting boundary-layer separation. For optimal performance he suggested, and was able to verify experimentally, that the boundary layer should be maintained as close as possible to separation, without actually separating (Stratford, 1959b). In this way, any specified pressure rise would be achieved in the shortest possible distance and with the least possible dissipation of energy. Thus, for example, an airfoil exhibiting this behaviour immediately after transition from laminar flow would be expected to have a very low drag.

Using boundary layer theory and the calculus of variations, Liebeck and Ormsbee (1970) developed a family of optimal pressure distributions. These utilized the Stratford pressure recovery distribution to provide maximum possible lift in an incompressible flow. An inverse airfoil design solution was then used to obtain the corresponding monoelement shapes. Liebeck (1973) then published the corresponding family of optimal velocity distributions, along with modifications necessary for obtaining realistic airfoil shapes. Using a new exact inverse airfoil design program, the corresponding airfoil shapes were determined and verified experimentally in wind tunnel tests. He has since designed and tested a variety of high lift airfoils for aircraft and other vehicles, such as racing cars (Liebeck, 1978, 1990).

In 1996, Venkataraman published two papers describing a design optimization approach to solving both direct and indirect design problems. In both cases, the airfoil geometry was defined with Bezier curves and the aerodynamic analysis was performed using a panel method. The optimizer used was an interactive software package called OptdesX, which uses a generalized reduced gradient method. The direct design problems (Venkataraman, 1996a) were to maximize the lift coefficient and to minimize the drag

coefficient. Using a Wortmann FX63-167 airfoil as an initial shape, he was able to find optimal shapes associated with a variety of Reynolds numbers and angles of attack. The inverse design technique (Venkataraman, 1996b) was demonstrated on a number of test cases for which solutions had already been obtained using other methods. Results showed excellent agreement with the target airfoils.

Chapter 3

Inverse Design Optimization Algorithm

The design optimization used to generate fan blade shapes in this work contains three elements. A geometry routine calculates a fan blade shape from a given set of control variables. This geometry is then passed to a flow solver, which calculates the pressure distribution around the given blade. An optimization routine searches for the best control variables - those that will minimize the difference between the required pressure distribution and that of the fan blade which is designed. For an exact solution, this quantity is expected to be zero. Here it is driven to a small number.

3.1 Flow Solver

An inviscid potential flow model is used for the flow calculations. Specifically, Martensen's surface vorticity panel method is used to model the flow through a turboma-

chinery blade cascade. Panel method theory has a long history of use in the aerospace industry and is well-defined and understood. The general theory behind the panel method can be found in many aerodynamics and fluid dynamics texts and therefore only a brief summary of the relevant theory and equations is given here.

3.1.1 Theoretical Background

Unlike other panel methods, the surface vorticity model is a direct simulation of physical reality. In all real flows around a body, adjacent to the surface there is a boundary viscous shear layer. Just outside this layer, the fluid will have a finite velocity, v_s. On the body surface, however, the fluid velocity is reduced to zero by the vorticity in the boundary layer. This condition of zero velocity on and parallel to the body surface is called the Dirichlet boundary condition. If the fluid viscosity could be reduced to zero, the thickness of this layer would become infinitesimal. If in addition the Reynold's number approached infinity, the boundary layer covering the body would be squashed into an infinitely thin vorticity sheet, $\gamma(s)$, where $\gamma(s)$ is defined as the vorticity strength per unit length at point s. The fluid velocity just outside this sheet is then exactly equal to the vorticity. That is, at any point s very near the surface of the body,

$$v_s = \gamma(s). \tag{3.1}$$

Consider first the two-dimensional flow past the cross-section of a single airfoil in the (x, y) plane. As suggested above, the flow is represented by the vorticity sheet $\gamma(s)$, initially of unknown strength. The arclength s is measured clockwise around the body perimeter, starting from the leading edge. Velocity at any point s_m will be induced by the

small vorticity elements, $\gamma(s_n)ds_n$, at all other points s_n on the body. Specifically, velocity dq_{mn} is given by the Biot-Savart law

$$dq_{mn} = \frac{\gamma(s_n)ds_n}{2\pi r_{mn}}. \tag{3.2}$$

where r_{mn} is the straight line distance between points s_m and s_n. The Dirichlet boundary condition can then be described at any point s_m by Martensen's boundary integral equation for two-dimensional flow,

$$-\frac{1}{2}\gamma(s_m) + \oint k(s_m, s_n)\gamma(s_n)ds_n + U_\infty \cos\rho_m + V_\infty \sin\rho_m = 0. \tag{3.3}$$

where $k(s_m, s_n)$ is derived from (3.2), (U_∞, V_∞) are the (x, y) components of the uniform stream W_∞, and ρ_m is the slope of the profile at s_m with respect to the x-axis.

The surface vorticity model is then represented discretely by choosing a finite number M of pivotal points (x_n, y_n). To accomplish this, M short, straight panels with endpoints (X_n, Y_n) are used to approximate the body surface. Element lengths are then

$$\triangle s_n = \sqrt{(X_{n+1} - X_n)^2 + (Y_{n+1} - Y_n)^2}. \tag{3.4}$$

The pivotal points are chosen at the centre of each panel,

$$\begin{cases} x_n = \frac{1}{2}(X_{n+1} + X_n) \\ y_n = \frac{1}{2}(Y_{n+1} + Y_n) \end{cases}. \tag{3.5}$$

The integral in (3.3) can then be evaluated using the trapezium rule. This results in

$$\sum_{n=1}^{M} K(s_m, s_n)\gamma(s_n) = -U_\infty \cos\rho_m - V_\infty \sin\rho_m, \tag{3.6}$$

which is a linear system of M equations (one for each pivotal point s_m) in the M unknown values of surface vorticity, $\gamma_n = \gamma(s_n)$.

Physically, the coupling coefficient $K(s_m, s_n)$ in (3.6) represents the velocity parallel to the body surface at s_m, induced by the vorticity along the nth panel. It is derived from equation (3.2):

$$K(s_m, s_n) = \frac{\triangle s_n}{2\pi} \frac{(y_m - y_n)\cos\rho_m - (x_m - x_n)\sin\rho_m}{(x_m - x_n)^2 + (y_m - y_n)^2}, m \neq n. \tag{3.7}$$

When $m = n$, the self-induced coupling coefficient is given by

$$K(s_m, s_m) = -\frac{1}{2} - \frac{1}{8\pi}(\rho_{m+1} - \rho_{m-1}). \tag{3.8}$$

Flow analysis for turbomachinery blade cascades is an extension of the model for single airfoils. A cascade is modelled as an infinite rectilinear array of airfoils set at equal pitch intervals t parallel to the y-axis, and with equal stagger angle λ (see Fig. 3.1). The flow is periodic in the y direction, so the nth element on all airfoils will have the same vortex strength $\gamma(s_n)\triangle s_n$. Consequently the velocity induced at element s_m becomes that induced by an infinite array of vortex elements each of strength $\gamma(s_n)\triangle s_n$. All that is necessary is to derive a modified coupling coefficient $K(s_m, s_m)$ to express this.

After a conformal transformation, it is found that the modified coupling coefficient is

$$K(s_m, s_n) = \frac{\triangle s_n}{2t}\left[\frac{\sin\left(\frac{2\pi}{t}(y_m - y_n)\right)\cos\rho_m - \sinh\left(\frac{2\pi}{t}(x_m - x_n)\right)\sin\rho_m}{\cosh\left(\frac{2\pi}{t}(x_m - x_n)\right) - \cos\left(\frac{2\pi}{t}(y_m - y_n)\right)}\right], m \neq n. \tag{3.9}$$

The self-inducing coupling coefficients for a cascade are in fact still given by (3.8). The same linear equations (3.6) as those used for the single airfoil may then be used to state the surface vorticity model for cascade flow, the modified coupling coefficient (3.9) being the only essential change.

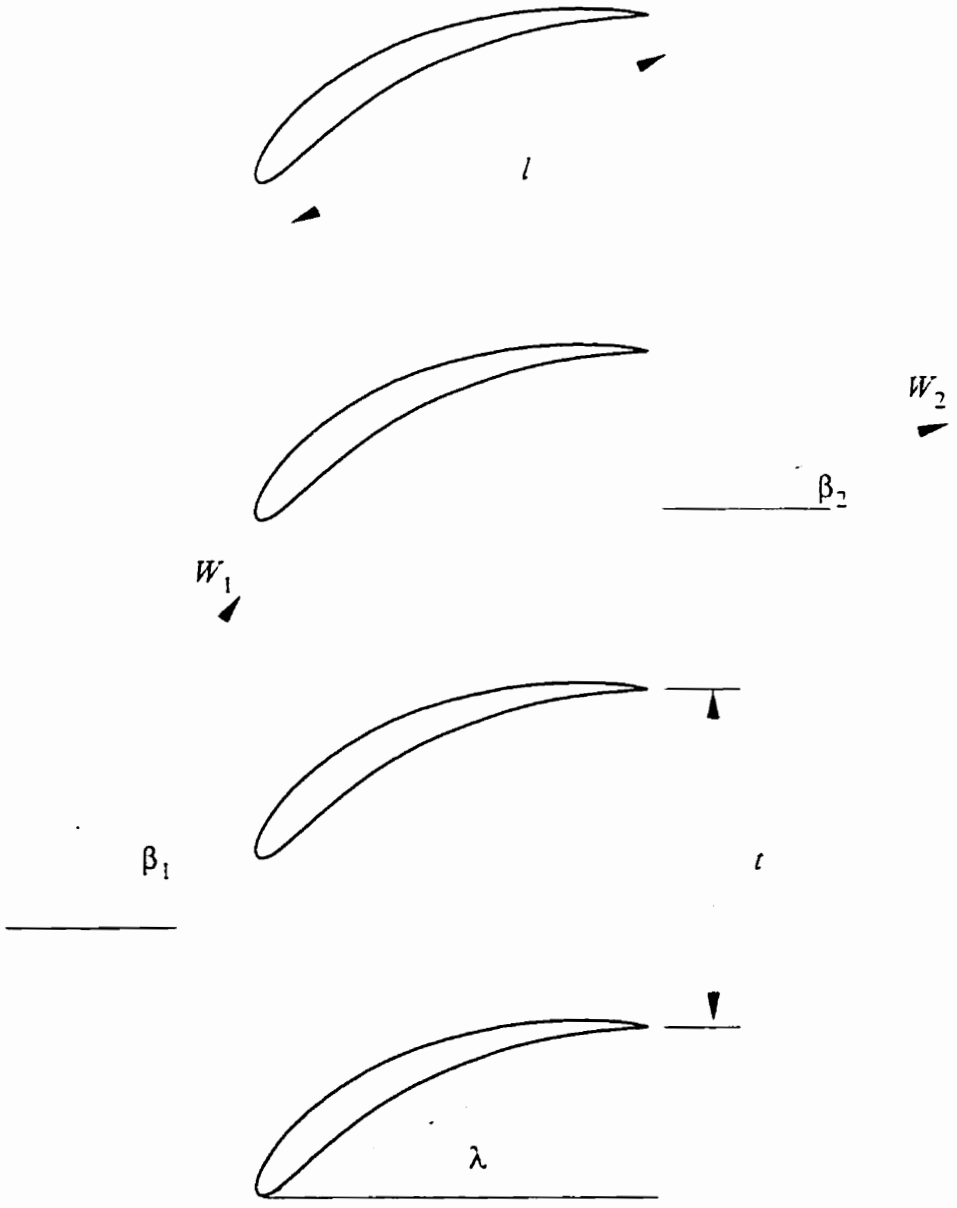


Figure 3.1: Cascade geometry and flow velocities.

In this configuration, however, the vorticities on opposite elements interfere with each other. This can be corrected by enforcing Kelvin's theorem in the flow system, which states that the net circulation around the interior of a profile must equal zero. The circulation induced around the profile interior due to a unit vortex located at s_m is approximated numerically as

$$\triangle\Gamma_m = \frac{1}{\triangle s_m} \sum_{n=1}^{M} K(s_n, s_m)\triangle s_n. \tag{3.10}$$

Enforcing Kelvin's theorem, (3.10) becomes

$$K(s_{M+1-m}, s_m) = -\frac{1}{\triangle s_{M+1-m}} \sum_{\substack{n=1 \\ n\neq M+1-m}}^{M} K(s_n, s_m)\triangle s_n. \tag{3.11}$$

Thus the coupling coefficient matrix should be corrected by replacing the back diagonal entries $K(s_{M+1-m}, s_m)$ with the value given in (3.11).

An adverse consequence of back diagonal correction is that any equation in the system is now equal minus the sum of all the others. That is, the matrix is now singular. For lifting bodies, the singularity may be corrected by using Wilkinson's trailing edge Kutta condition. The Kutta condition states that the surface vorticity at the two trailing edge elements should have the same magnitude. That is, the constraint

$$\gamma(s_{te}) = -\gamma(s_{te+1}) \tag{3.12}$$

should be imposed.

After imposing the Kutta condition, since $\gamma(s_{te})$ and $-\gamma(s_{te+1})$ are equal, there is now one less unknown value. That is, introducing (3.12) into the linear system (3.6), the nth equation may be written with $M-1$ unknowns

$$K_{n,1}\gamma_{s_1} + ... + [K_{n,te} - K_{n,te+1}]\,\gamma_{te} + ... + K_{n,M}\gamma_M = rhs_n, \tag{3.13}$$

where $K_{i,j} = K(s_i, s_j)$, $rhs_n = -U_\infty \cos\rho_n - V_\infty \sin\rho_n$, and $\gamma_i = \gamma(s_i)$. Wilkinson (1967) suggested that, since the number of unknowns has been reduced, one equation may also be eliminated. To do this, the most effective technique is to subtract the two equations representing the trailing edge elements. Thus, Wilkinson's trailing edge Kutta condition not only enforces the Kutta condition, but also reduces the linear system by one equation and one unknown. This resolves the singularity created by back diagonal correction.

3.1.2 Code Development

The potential flow model used by the surface vorticity method simplifies the true fluid dynamics situation. It will, however, accurately predict the flow in most situations. Its biggest advantage in an optimization scheme is the speed with which it can perform the calculations. The code is translated from the Pascal code given by Lewis (1991) into a self-contained C++ function named, appropriately, FlowSolver(). The flow chart in Fig. 3.2 shows the main computational stages. The input and output parameters and each main computational stage are described in this section.

The input and output is done through function parameters instead of files. This facilitates integration with the rest of the aerodynamic shape design package. The input parameters are the number of profile data points M, the pitch/chord ratio t/l, the stagger angle λ, the inlet velocity W_1, and the inlet angle β_1 (see Fig. 3.1). All angles are input in degrees.

Certain requirements must be imposed on the distribution of panel endpoints to ensure accurate modelling of the flow. These requirements will have to be met by the routine generating the geometry. First, the profile data points on the upper and lower

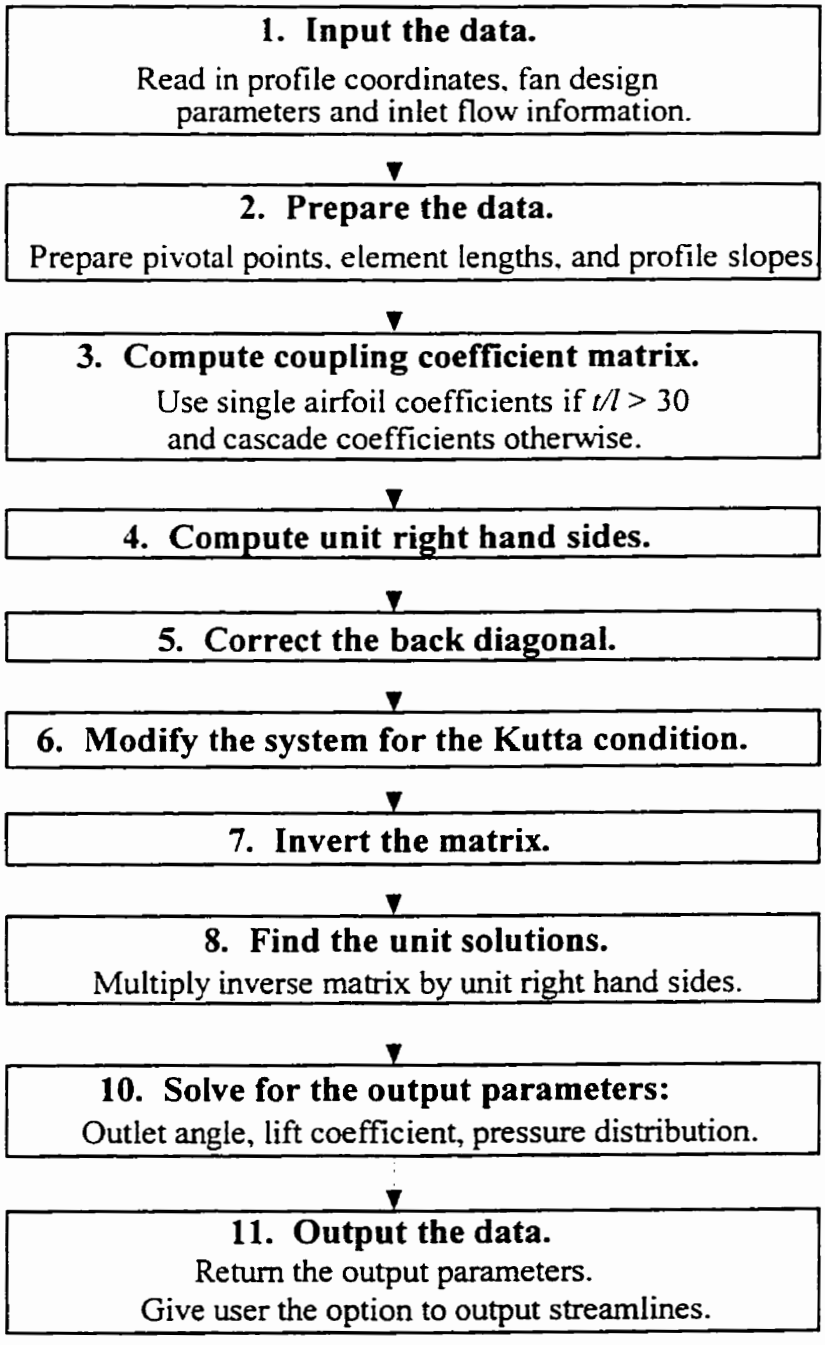


Figure 3.2: Flow chart for the surface vorticity panel method.

surfaces of the blade should lie directly opposite one another in pairs. That is, they should lie along the same normal to the profile camber line. Second, the panels should be selected in such a way that $\Delta s_m \leq \Delta T_m$, where Δs_m is the element length, and ΔT_m is the local profile thickness. In fact, the parameter $\Delta T_m / \Delta s_m$ is of greater significance than simply the total number of elements. Finally, the number of panel endpoints specified must be odd, so that there are an even number of panels. It is worth noting that the first endpoint in the list must equal the last endpoint, ensuring profile closure.

The function's output parameters include the pressure distribution (surface pressure coefficient C_p on each panel), the lift coefficient C_L, the blade bound circulation Γ, and the outlet angle β_2. The boolean parameter Optimized is also passed to the routine. Optimized = TRUE indicates that a solution to the design problem has been found. In this case, the flow solver outputs the pressure distribution to a file and gives the user the opportunity to output the streamlines to a file.

After the data is input, it is prepared for the computations. Element lengths are given by (3.4) and equation (3.5) is used to find the pivotal points. The profile slopes satisfy the finite difference equations

$$\begin{cases} \cos \rho_n = (X_{n+1} - X_n)/\Delta s_n \\ \sin \rho_n = (Y_{n+1} - Y_n)/\Delta s_n \end{cases} . \tag{3.14}$$

These values are then used to evaluate the profile slope ρ_n, taking care to select the correct quadrant. Lewis (1991) is not quite careful enough, and a small modification was made to his code at this point. The coupling coefficient for any element requires the difference between profile slopes of the adjacent elements, $(\rho_{n+1} - \rho_{n-1})$. (See equation (3.8).) To do this properly, the code was modified to ensure that $0 \leq \rho_n \leq \pi$ for all panels on the upper

half of the blade, that is, those numbered previous to the trailing edge, `te`. Then, as the airfoil is traversed clockwise from the leading edge, the slopes decrease where the profile is convex, and increase where it is concave. By contrast, when Lewis's code encounters an airfoil shape with a leading edge angle greater than $\pi/2$, as in Fig. 3.3b, it incorrectly calculates a negative ρ_1. Then, for example, if the third profile slope ρ_3 is less than $\pi/2$, it is calculated correctly by Lewis, implying that $(\rho_3 - \rho_1)$ will be π too large. Thus the Pascal code fragment from Lewis

```
if cosine[n]<0 then slope[n]=t-pi;
```

was replaced by the C code

```
if (cosine[n]<0) {if (n<te) slope[n]=t+pi; else slope[n]=t-pi;}
```

Note that the slopes calculated here depend only on the airfoil itself, ignoring the stagger angle of the cascade.

The coupling coefficient matrix for the cascade is then computed. The self-inducing coupling coefficients K_{mm} are calculated using equation (3.8), which is the same as the formula for a single airfoil. If the cascade has very wide blade spacing, equation (3.7) for single airfoils is used to compute the non-self-inducing coupling coefficients $K_{mn}, m \neq n$. Otherwise, equation (3.9) is used for the modified cascade coupling coefficients.

Fourth, the right hand side vectors of the linear systems are found. Instead of using (3.6), two unit vectors are used, corresponding to $U_\infty = 1.0$ and $V_\infty = 1.0$. Then the linear system (3.6) can be split into the two linear systems

$$\begin{cases} \sum_{n=1}^{M} K(s_m, s_n)\gamma_u(s_n) = -\cos\rho_m \\ \sum_{n=1}^{M} K(s_m, s_n)\gamma_v(s_n) = -\sin\rho_m \end{cases}, \tag{3.15}$$

where $\gamma_u(s_n)$ and $\gamma_v(s_n)$ are unit vorticities defined by

$$\gamma(s_n) = U_\infty \gamma_u(s_n) + V_\infty \gamma_v(s_n). \tag{3.16}$$

In this way the flow through the cascade corresponding to any inlet angle and velocity can be calculated economically after solving the unit system.

Back diagonal correction of the coupling coefficient matrix K_{mn} enforces Kelvin's theorem (3.11), and then Wilkinsons's trailing edge Kutta condition is applied (3.13). This eliminates one unknown value of surface vorticity, so one equation must also be eliminated. Rows te and $te + 1$ of the matrix K and of the right hand side vectors are replaced with the difference between them, reducing the matrix size by one.

The system is now ready to be solved. The seventh stage finds the inverse, K^{-1}, of the coupling coefficient matrix. This matrix has finite values throughout with a dominant leading diagonal, offering no difficulties for solution by matrix inversion. The algorithm coded needs very little extra memory, making it ideal for use on a personal computer. Also, since there are two right hand side vectors, the same inverse matrix is used to solve both systems, halving the computational time.

The unit solutions, $\gamma_u(s_n)$ and $\gamma_v(s_n)$, are then found using the matrix multiplications $\overrightarrow{\gamma_u} = K^{-1}\overrightarrow{rhs1}$ and $\overrightarrow{\gamma_v} = K^{-1}\overrightarrow{rhs2}$. Since the system has been reduced in size by one, the true unit solutions must be recovered by splitting the trailing edges using (3.12) and shifting the lower edge vector elements down by one. At this stage, the unit bound circulations (that is the unit bound vortex strengths), Γ_u and Γ_v, are calculated using

$$\begin{cases} \Gamma_u = \sum_{n=1}^{M} \gamma_u(s_n)\Delta s_n \\ \Gamma_v = \sum_{n=1}^{M} \gamma_v(s_n)\Delta s_n \end{cases} . \tag{3.17}$$

The solution for the flow is then found. First, the outlet flow angle, β_2, is calculated using

$$\beta_2 = \arctan\left\{\left(\frac{1-\Gamma_v/2t}{1+\Gamma_v/2t}\right)\tan\beta_1 - \left(\frac{2}{1-\Gamma_v/2t}\right)\frac{\Gamma_u}{2t}\right\}. \tag{3.18}$$

Then β_∞ is calculated from β_1 and β_2 using

$$\tan\beta_\infty = \frac{1}{2}\left(\tan\beta_1 + \tan\beta_2\right). \tag{3.19}$$

The velocity at infinity, W_∞, is the vector mean of the inlet and outlet velocities. Its components, U_∞ and V_∞, are then found. The surface pressure distribution,

$$C_p = 1 - \left(\frac{\gamma(s)}{W_1}\right)^2. \tag{3.20}$$

is typically plotted against x/l, where x is the x-coordinate of the profile, and l is the chord length. Both parameters are computed at each pivotal point. The blade bound circulation, Γ, is found using U_∞, V_∞, and the unit bound circulations, Γ_u and Γ_v. Finally, the lift coefficient, C_L, is calculated using

$$C_L = \frac{2\Gamma}{W_\infty l}. \tag{3.21}$$

3.1.3 Testing and Verification

The panel method flow solver was verified using a number of different tests. A test using an offset circle was performed to verify the correction to Lewis's code described above. Then the full flow solver was tested on three fan blade shapes for which exact solutions are known and given by Lewis (1991). Finally, the streamline computation was tested on an elliptical shape.

As noted above in section 3.1.2, a correction was made to the data preparation routine given in Lewis (1991). Lewis's formula for calculating the profile slope, ρ_n, is

incorrect when the first slope is greater than $\pi/2$. To verify the correction, the data preparation routine was tested for the flow past a circular cylinder, for which the exact solution is known. A first test shows that both slope calculations are in agreement when $\rho_1 < \pi/2$. A second test shows that the modified formula is correct when $\rho_1 > \pi/2$.

If coordinates of the cross-section of a circular cylinder of radius a are expressed by

$$\begin{aligned} x &= a(1 - \cos\phi) \\ y &= a\sin\phi \end{aligned}$$

then the exact solution for the surface velocity due to a uniform stream U_∞ parallel to the x-axis, Batchelor (1970), is

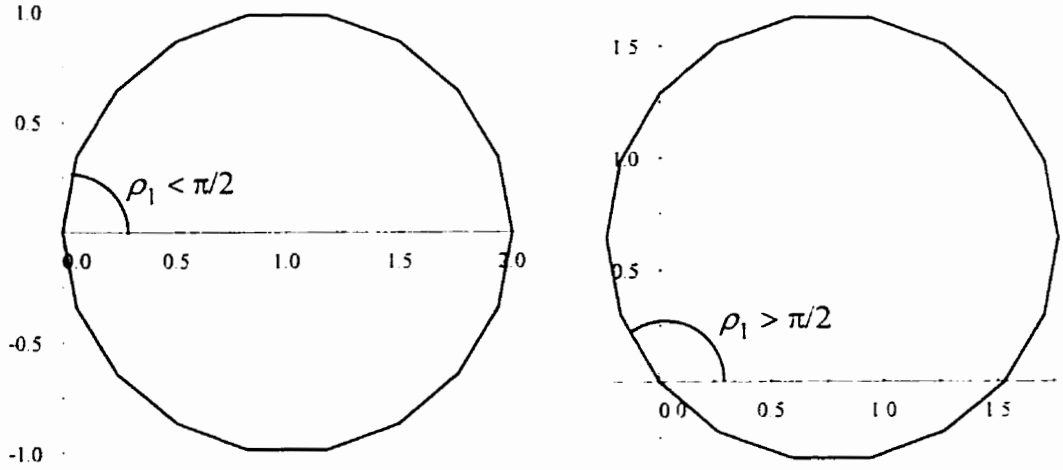
$$v_s = 2U_\infty \sin\phi.$$

In the first test, the surface velocity is compared for a circle of radius $a = 1$, with diameter along the x-axis, and 18 elements (Fig. 3.3a). Starting at $(0,0)$, the panels are numbered clockwise around the circle. As shown in Fig. 3.3a, $\rho_1 < \pi/2$. Table 3.1 gives velocity information for the first 9 elements, verifying that the code correctly calculates the velocities when both angle formulae are used.

In the second test, the same circle is used ($a = 1$ with 18 panels), but is offset from the x-axis by using coordinates

$$\begin{aligned} x &= (1 - \cos\phi) - \left(1 - \cos\frac{2\pi}{9}\right) \\ y &= \sin\phi + \sin\frac{2\pi}{9} \end{aligned}$$

(see Fig. 3.3b). Again the panels are numbered clockwise starting at the origin, so that



a. Circle centered on axis

b. Circle offset from axis

Figure 3.3: Circle test used to verify a modification to Lewis's code.

Panel Number	v_s exact	v_s Lewis	v_s with slope correction
1	0.347296	0.347098	0.347098
2	1.000000	0.999429	0.999429
3	1.532089	1.531214	1.531214
4	1.879385	1.878311	1.878311
5	2.000000	1.998856	1.998856
6	1.879385	1.878311	1.878311
7	1.532089	1.531214	1.531214
8	1.000000	0.999429	0.999429
9	0.347296	0.347098	0.347098

Table 3.1: Potential flow past a circle with diameter on x-axis.

Panel Number	v_s exact	v_s Lewis	v_s with slope correction
1	-1.000000	1.361130	-0.999429
2	-0.347296	0.889026	-0.347098
3	0.347296	1.351910	0.347098
4	1.000000	2.679810	0.999429
5	1.532089	3.211600	1.531214
6	1.879385	3.558700	1.878311
7	2.000000	3.679240	1.998856
8	1.879385	3.558700	1.878311
9	1.532089	3.211600	1.531214

Table 3.2: Potential flow past a circle offset from the x-axis

$\rho_1 > \pi/2$. Table 3.2 demonstrates the error in Lewis's code, and the validity of the correction to the slope formula.

It should be noted that this correction is only necessary when a portion of the leading edge of the airfoil is plotted in the second quadrant. This situation will occur when camber and thickness curves are used to generate airfoil shapes, which is the technique used in this work. For example, examine the leading edge of the C4/70/C50 profile (Fig. 3.4a), tested below. However, the region in which Lewis's slope formula is wrong is usually quite small, so the correction does not typically affect the velocity distribution as significantly as it does in the offset circular cylinder above.

Having verified the modifications to the data preparation routine, the entire flow solver subroutine was then tested using two profiles given by Lewis. In each case, there is agreement in the shape of the pressure distribution, and in the outlet angle calculated.

The first full test is a compressor blade with a C4 base profile distributed upon a 70° circular arc camber line. Lewis provides the profile coordinates and flow results for a pitch/chord ratio of 0.900364 and zero stagger angle, with inlet angles β_1 of ±35°. The

C4/70/C50 profile with $\lambda = 0$, $t/l = 0.900364$			
Method		$\beta_1 = +35°$	$\beta_1 = -35°$
Exact solution, Gostelow (1984)	$\beta_2 =$	-23.80	-24.84
Subroutine FlowSolver()	$\beta_2 =$	-23.85	-25.04

Table 3.3: Comparison of Martensen's method with exact cascade theory - Test 1

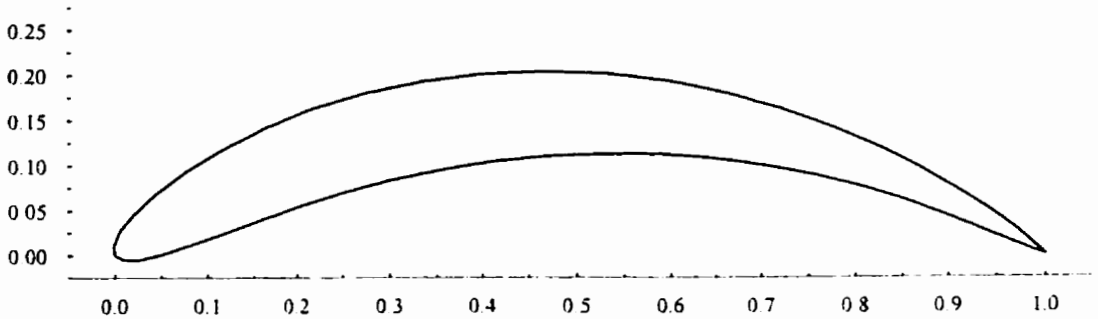
112° Camber profile with $\lambda = 0$, $t/l = 0.5899644$, $\beta_1 = 50°$	
Method	β_2
Exact solution, Gostelow (1984)	-24.84
Subroutine FlowSolver()	-25.04

Table 3.4: Comparison of Martensen's method with exact cascade theory - Test 2

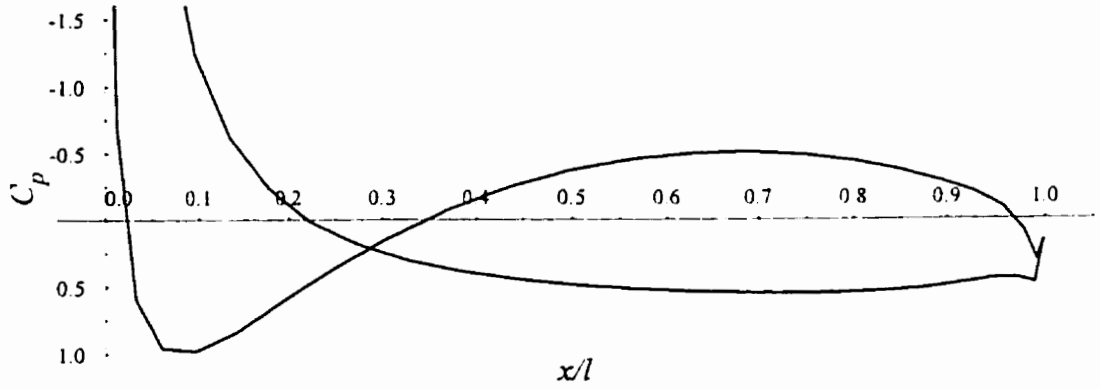
C4/70/C50 profile and pressure distributions computed by FlowSolver() are shown in Fig 3.4. The distributions compared favourably with the exact solution given by Gostelow (1984). In Table 3.3 the outlet angle, β_2, generated by the code is compared with that of the exact solution.

The second test is the highly cambered (112°) impulse cascade profile shown in Fig. 3.5a. Lewis provides the profile coordinates and flow results for a pitch/chord ratio of 0.5899644 and zero stagger angle, with an inlet angle of 50°. The pressure distribution shown in Fig. 3.5b demonstrated excellent agreement with the exact solution, again given by Gostelow (1984). In Table 3.4 the outlet angle, β_2, generated by the code is compared with that of the exact solution.

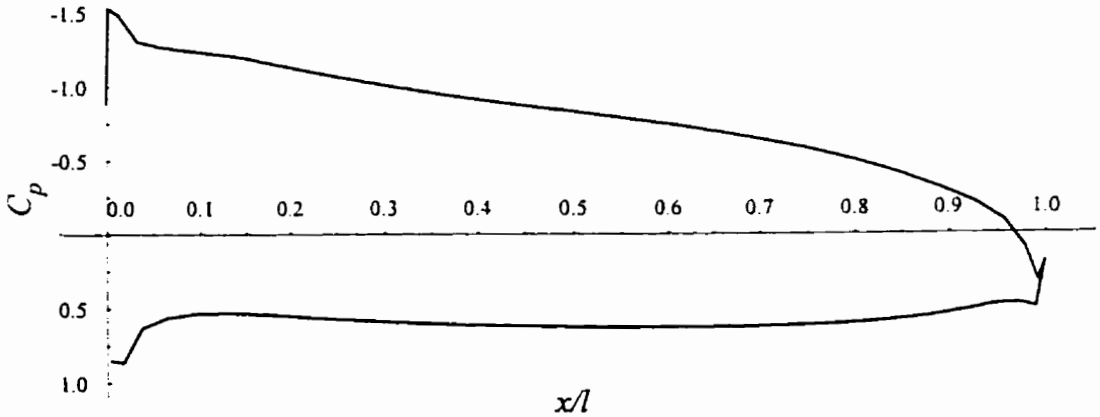
The third test is a Joukowski airfoil typical of an inlet guide vane. Lewis provides the profile coordinates and flow results for a pitch/chord ratio of 1.549687, stagger angle of -10.0992°, and inlet angle of 0°. In Table 3.5 the outlet angle, β_2, generated by the code is compared with that of the exact solution, given by Lewis. Although not shown for this



a. C4/70/C50 fan blade shape

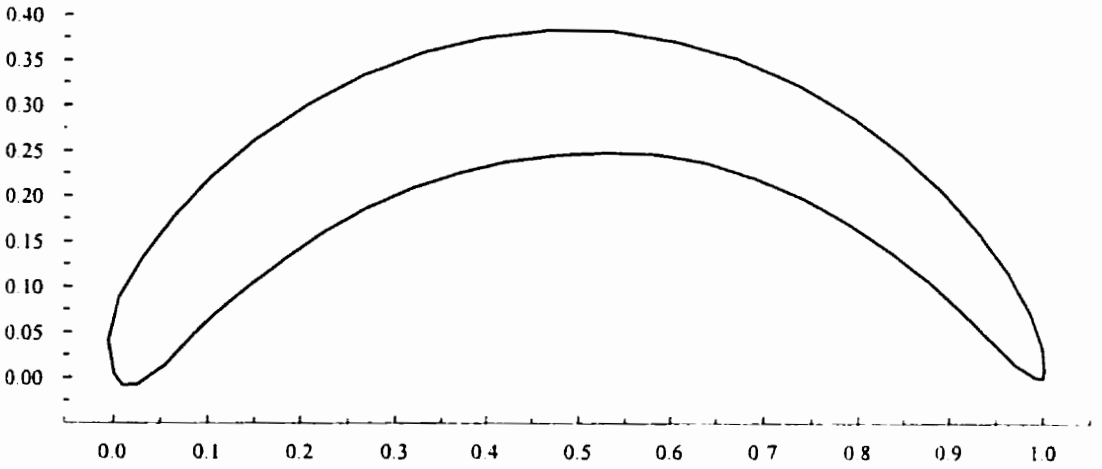


b. Pressure distribution $\beta_1 = -35^\circ$

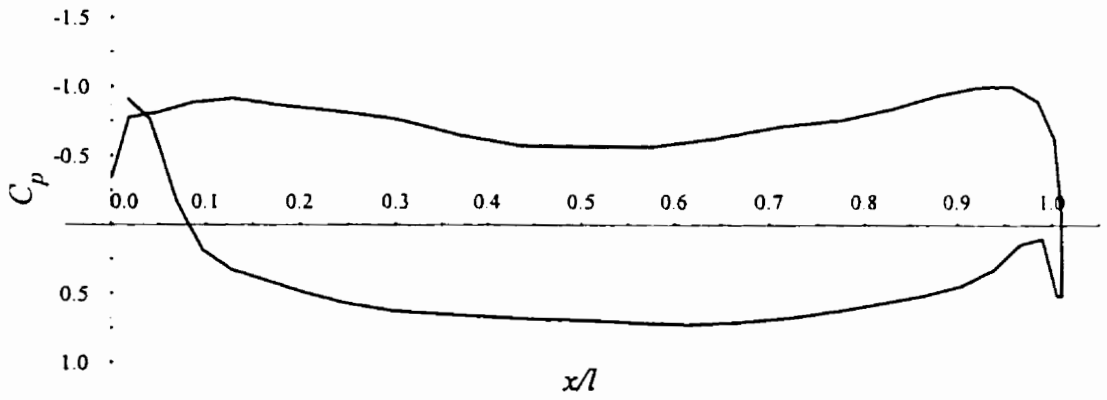


c. Pressure distribution $\beta_1 = +35^\circ$

Figure 3.4: Test of the flow solver using profile C4/70/C50.



a. Highly cambered fan blade shape



b. Pressure distribution

Figure 3.5: Test of the flow solver using a 112° cambered profile.

Inlet Guide Vane with $\lambda = -10.0992°$, $t/l = 1.549687$, $\beta_1 = 0°$	
Method	β_2
Exact solution, Lewis (1991)	-30.0024
Subroutine FlowSolver()	-29.9791

Table 3.5: Comparison of Martensen's method with exact cascade theory - Test 3

case, there was once again excellent agreement between the pressure distribution published in Lewis and that generated by the FlowSolver() code.

3.2 Shape Definition

Bezier curves are a useful tool in optimization. They can be defined using relatively few parameters, are easily constrained, and do not oscillate wildly. The purpose here is to use Bezier curves to generate an airfoil mesh that has as few elements as possible while still allowing the flow to be computed accurately.

3.2.1 Theoretical Background

Phillip Bezier formulated his polynomial curves to assist in the design of sculptured surfaces (Bezier, 1974). A Bezier curve is uniquely determined by vertices of a polygon, called the control points. These define the derivatives, order, and shape of the curve, but only the first and last control points actually lie on the curve. The polynomials used for the curve are of degree one less than the number of control points.

A Bezier curve is defined as the following parametric curve $\mathbf{P}(u)$ in terms of $n+1$ control points $\mathbf{P}_i$ (Bartels et al., 1987):

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,n}(u),\ 0 \leq u \leq 1, \tag{3.22}$$

where $B_{i,n}(u)$ is the blending function given by:

$$B_{i,n}(u) = C(n,i)u^i(1-u)^{n-i}. \tag{3.23}$$

$C(n,i)$ is the binomial coefficient:

$$C(n,i) = \frac{n!}{i!(n-i)!}. \tag{3.24}$$

Thus, if the two-dimensional control points are $\mathbf{P}_i = (x_i, y_i)$, then the curve $\mathbf{P}(u)$ can be found parametrically from:

$$\begin{cases} x(u) = \sum_{i=0}^{n} x_i B_{i,n}(u), \\ y(u) = \sum_{i=0}^{n} y_i B_{i,n}(u). \end{cases} \tag{3.25}$$

Bezier curves have many properties that make them attractive for defining the airfoil surface during the design procedure as used here (Newman & Sproull, 1979). The end points are automatically fixed at the two end vertices. The vector joining the endpoint and the closest control point is tangent to the curve at its endpoint. The curve always lies within the convex figure defined by the extreme points of the polygon. Finally, the curve is nth order continuous throughout and never oscillates wildly away from its defining control points.

To define a general airfoil shape with as few control points as possible, four Bezier curves will be joined, following Venkataraman (1995a). He used two curves to define the upper surface of a wing's shape, and two for the lower surface. In contrast to Venkataraman, however, upper and lower surface curves will not be used in this work for reasons that follow.

One of the mandates of the shape definition module is to generate a mesh with the least possible number of panels, M. Since the flow solver solves an $M \times M$ linear system, this will minimize the runtime for a flow calculation. Of course, choosing too few panels

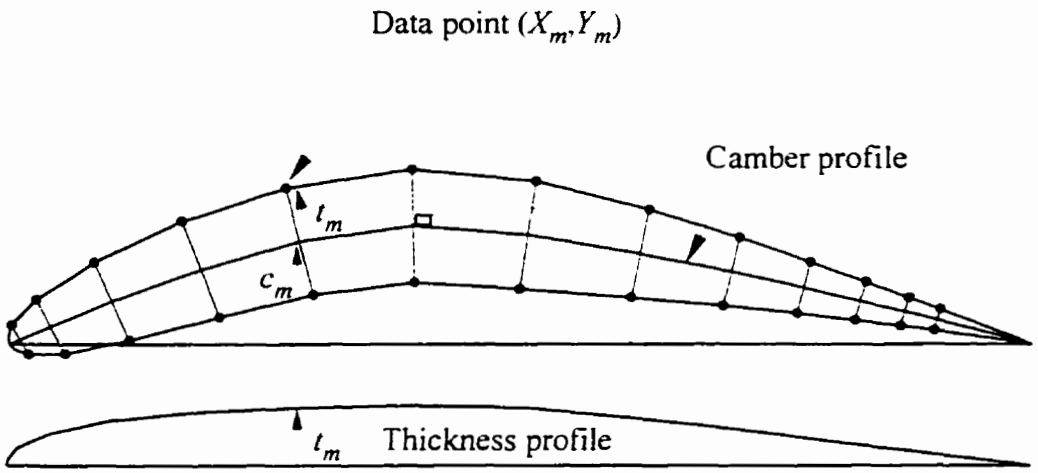


Figure 3.6: Superposition of thickness normal to camber to generate an airfoil shape.

would compromise the validity of the result. Recall, then, that for the vorticity method to compute the flow accurately, each panel length, $\triangle s_m$, should not be greater than the local thickness of the blade, $\triangle T_m$. Maintaining the ratio $\triangle T_m / \triangle s_m = 1$ will thus maximize the length of each element, minimizing the total number of elements.

It is common to define airfoils using thickness and camber profiles. The local thickness at any point, $\triangle T_m$, can most easily be determined by using this configuration. Thus two Bezier curves are joined to form the half thickness profile, and two Bezier curves form the camber profile. The airfoil shape is then generated in the standard manner by superposing the half thickness normal to the camber, as shown in Fig. 3.6. This superposition further satisfies the remaining significant requirement of the vorticity method's mesh, namely that upper and lower surface data point pairs should lie along the same normal to the camber curve.

Some geometric constraints on the control points will be required to ensure a

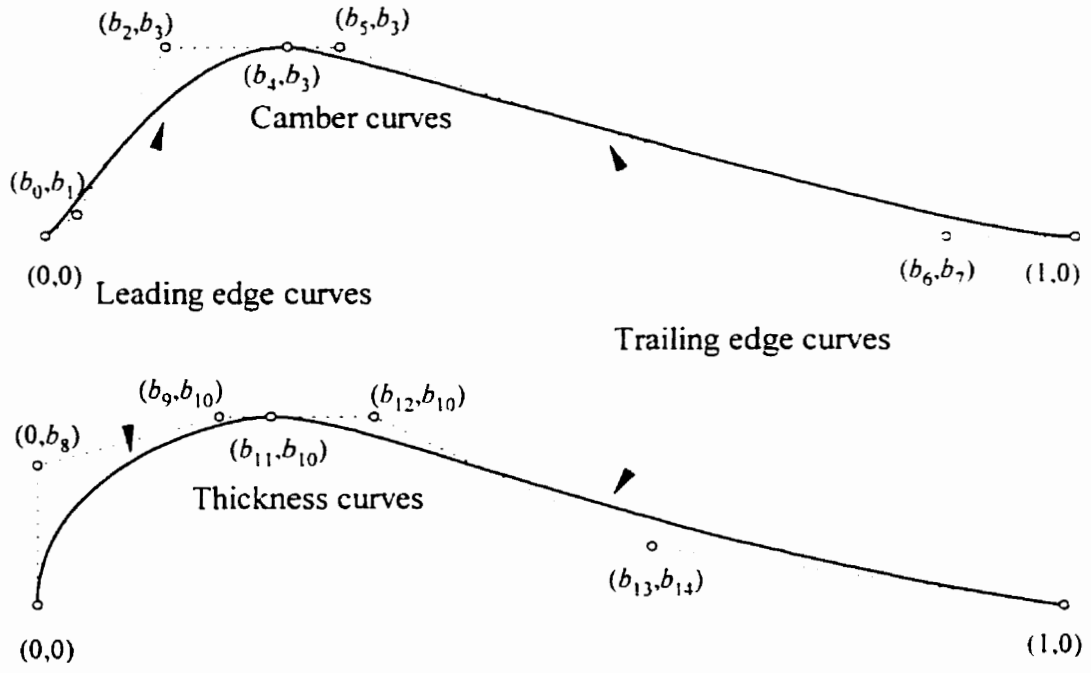


Figure 3.7: Bezier control variables required to form an airfoil shape.

smooth airfoil shape (refer to Fig. 3.7). A subsonic airfoil should have a rounded nose. This is achieved by constraining the first polygon edge on the leading thickness curve to be normal to the x-axis. At the joining points between the leading and trailing Bezier curves, continuity is obtained by forcing the end control points to coincide. The derivative is made continuous by insisting on horizontal colinearity of the two polygon edges adjacent to these common endpoints. Insisting that the colinear edges be horizontal allows the total number of degrees of freedom of the problem to be decreased by two, one for each of the thickness and camber profiles. It will also simplify the determination of the maximum camber and thickness values of designed blades.

These geometric constraints are demonstrated in Fig. 3.7 for cubic leading and

trailing curves. The variables labelled b_i, $i = 0, ..., 14$ in the figure will be called the Bezier control variables, and are the only variables needed to define the shape. A further constraint has been added to simplify the problem: the thickness and camber curves to start at (0,0) and end at (0,1). Notice that to increase by one the order of any of the four curves, one Bezier control point will be added, requiring two additional Bezier control variables b_i.

An important feature of parametrizing the shape with Bezier curves is that perturbations of the Bezier vertices will still define airfoil shapes. It is also significant that further geometric constraints could easily be imposed to force the designed shape to conform to specific structural and manufacturing needs. For example, a designer might require the maximum thickness to occur at 25% of the chord in order to maximize the strength of the beam at the twisting moment. The variable b_{11} joining the thickness curves would then be set equal to 0.25.

3.2.2 Code Development

The shape definition code is written as a self-contained C++ function, Geometry(). It first tests the Bezier control variables for violation of any constraints. If the variables pass the test, it uses Bezier curves to compute the camber and thickness profiles, and then the data points defining the blade. The flow chart in Fig. 3.8 shows the overall structure of the subroutine. The routines actually used to calculate the points on the Bezier curves are very straightforward and were coded directly from equations (3.23), (3.24), and (3.25).

One input parameter of the Geometry() function is the array of Bezier control variables, b_i. These must be ordered precisely as in Fig. 3.7. The boolean flag Optimized is also passed to the routine. If Optimized = TRUE, the data points and control points are

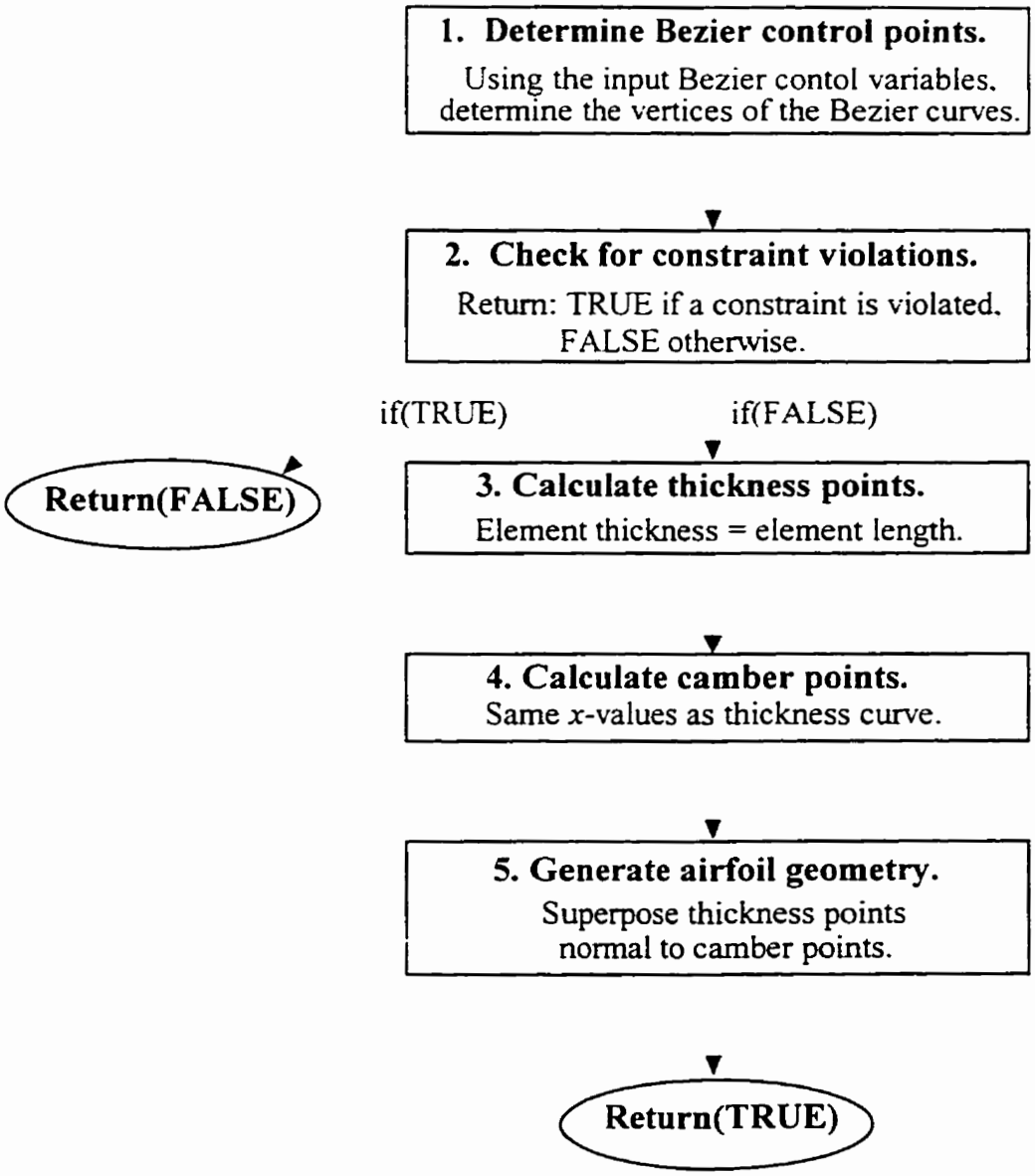


Figure 3.8: Flow chart for the shape definition module.

output to files. The output parameter is the array of fan blade data points. The function also has a boolean return value. TRUE is returned if the routine was successfully able to compute the data points without any constraint violations. FALSE is returned otherwise.

The first function call determines the actual Bezier control points from the list of control variables, b_i. The function separates the thickness control points from the camber control points. The relationship between the Bezier vertices and control variables can be seen in Fig. 3.7. While that figure only depicts cubic curves, the code is more general. The global variables NLP and NTP define, respectively, the number of vertices used to define the leading and trailing edge curves. Each value is typically four, but when NLP is increased, for example, another Bezier vertex is inserted in both leading edge curves. This will require two additional control variables on each curve, increasing the number of degrees of freedom by four.

The Bezier control points are then checked for any constraint violations. Some of these constraints must always be imposed, such as positive y-values on the thickness curve. Most constraints are problem-dependent, however, and can be easily added by the designer. Constraints developed during the testing stage of the algorithm are described below in section 3.2.3. If any constraint is violated, Geometry() immediately returns a FALSE value. This facilitates the use of a penalty function in optimization, which generates large errors for airfoils which violate any constraint. This penalty function is described below in section 3.4.3. By using it, the optimization code avoids many needless calculations of meaningless constraint-violating shapes and their corresponding potential flows.

The thickness profile points are computed next. These points will follow the

Bezier curve determined by the thickness control points. As noted above, the data points are determined such that the local thickness $\triangle T_m$ is maintained approximately equal to the element length $\triangle s_m$. $\triangle T_m$ is just the second component, t_m, of the Bezier thickness curve at the point (x_m, t_m). $\triangle s_m$ will be approximated by the corresponding element length on the thickness curve. That is, when one point (x_m, t_m) is known, the next point on the curve, (x_{m+1}, t_{m+1}), will be chosen such that

$$\sqrt{(x_{m+1} - x_m)^2 + (t_{m+1} - t_m)^2} = y_m. \tag{3.26}$$

The Bezier parameter, u, is sometimes referred to as the normalized arclength of the Bezier curve. This misleading definition led to the idea of using the arc length between consecutive thickness points to approximate the element length. However, the parameter is in fact not a measure of a Bezier curve's arclength in general. Linear interpolation is used instead to satisfy equation (3.26). To avoid an unreasonably coarse mesh in very thick regions, the constant Lmax=0.05 was introduced to define a maximum element size.

Now that the points on the thickness profile have been computed, the camber profile is considered. To be consistent with definitions in the aerodynamic engineering community, the x-values along the camber curve should be the same as those along the thickness curve. Once again, interpolation is the technique used to find the parameters u_m for which the x_m values on the Bezier camber curves match those on the thickness profile. The Bezier parameters u_m are then used to find the camber values c_m.

Having defined the thickness points (x_m, t_m) and camber points (x_m, c_m), it remains only to compute the data points that will form the fan blade shape. The superposition of the thickness profile normal to the camber profile is accomplished using the following

formula:

$$\begin{cases} X_m = x_m \pm t_m \sin(\theta_c(x_m)) \\ Y_m = c_m \pm t_m \cos(\theta_c(x_m)) \end{cases} . \tag{3.27}$$

where $\theta_c(x_m)$ is the slope of the camber profile. This defines the fan blade data points (X_m, Y_m) which are passed out of the geometry routine.

3.2.3 Geometric Constraints

Several constraints are imposed on the Bezier control points in the generation of the geometry. These can be categorized as either implicit or explicit. The implicit constraints are those depicted in Fig. 3.7, including horizontal colinearity at the curve junctures and normality of the thickness curve at the origin. These allow a reduction in the number of degrees of freedom of the optimization problem. If four control points are used on each of the leading and trailing edge curves, fifteen degrees of freedom are necessary to define the geometry.

The explicit constraints are coded into the function ConstrainCtrlPts(). These can be determined experimentally, or may impose specific structural and manufacturing requirements. Perhaps the most obvious constraint on the thickness curve is that it remain positive throughout. The constraint is imposed by insisting that all y-values of the thickness control points be positive, except at the endpoints (0,0) and (1,0).

Other explicit constraints imposed on both camber and thickness control points are: x-values lie between 0 and 1; x-values are all distinct on a given curve; control points are ordered left to right (i.e. downstream); x-values of colinear points at the juncture of the curves are separated by at least Sep=0.05. All of these constraints are used to ensure

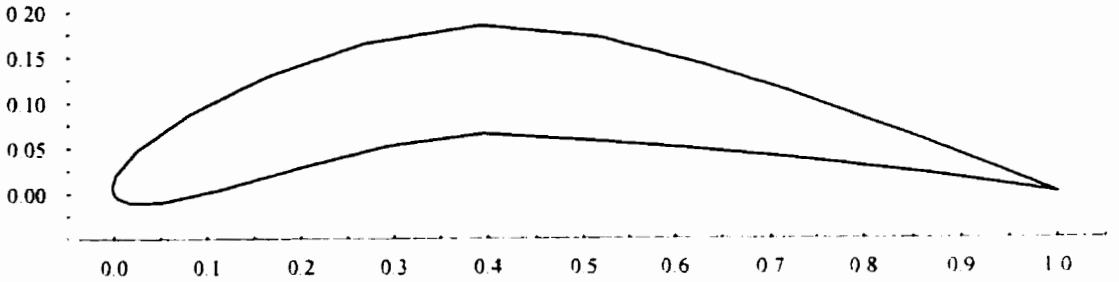
the resulting Bezier curves will avoid sharp bends and will not be multivalued.

3.2.4 Testing and Verification

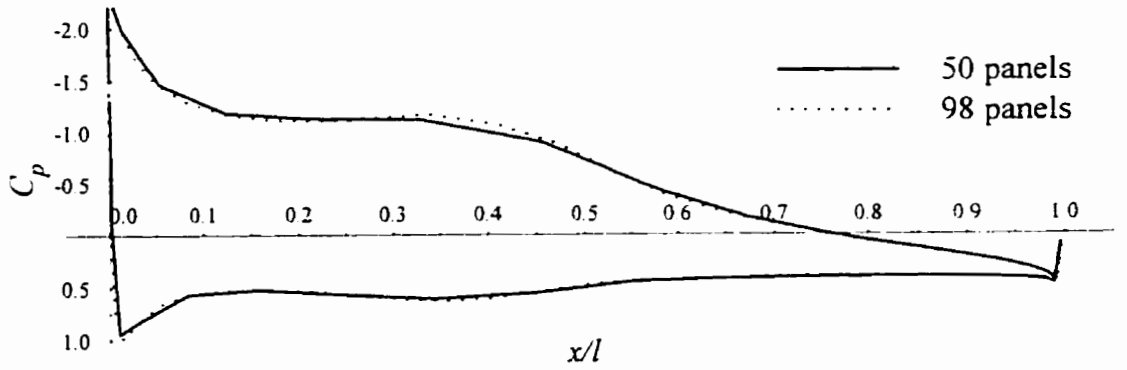
Several tests were run using Bezier control variables that seemed to generate realistically shaped fan blades. The calculated data points were input to the flow solver to determine the corresponding pressure distribution. The code was then used to generate a shape with a finer mesh. This was then used to test the accuracy of the original flow calculation. All tests verified the Bezier curve generation. The technique of determining the position of the data points was found to be very effective at minimizing the number of panels necessary for the flow solver to produce correct results. Only two tests are described here.

Using the control points in Table 3.6, airfoil shape Bez4a4f (Fig. 3.9a), with a maximum thickness of 12% of the chord, was generated with 50 panels when using $\Delta s_m/\Delta T_m = 1$. The mesh was then refined by using $\Delta s_m/\Delta T_m = 1/2$, generating 98 panels. Both meshes were passed to the flow solver to compare the resulting pressure distribution. For fan parameters $\beta_1 = 35°$, $\lambda = 0$, $t/l = 0.900364$, $W_1 = 1.0$, the pressure distributions shown in Fig. 3.9b were calculated. The agreement between the results confirms the sufficiency of the ratio $\Delta s_m/\Delta T_m = 1$. Note that this test also demonstrates the ability to have different joining points on the thickness and camber curves. Here the Bezier thickness curves join at $x = 0.45$. The Bezier camber curves join at $x = 0.4$.

For the second test, using the control points in Table 3.7, airfoil shape Bez5a4f (Fig. 3.10a), with a maximum thickness of 8% of the chord, was generated with 60 panels when using $\Delta s_m/\Delta T_m = 1$. The mesh was then refined by using $\Delta s_m/\Delta T_m = 1/2$, generating



a. Fan blade shape

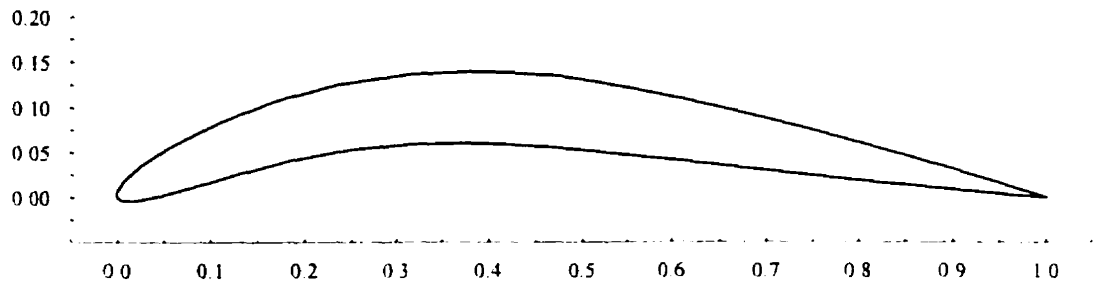


b. Pressure distributions

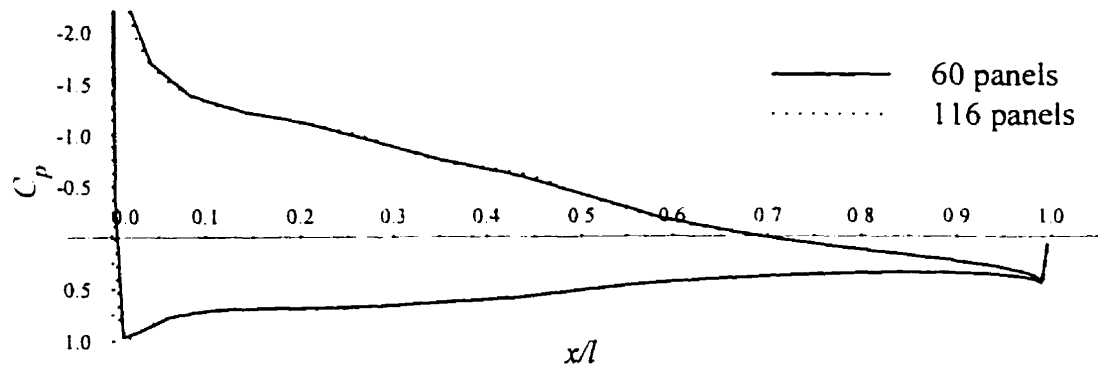
Figure 3.9: Geometry test: airfoil Bez4a4f.

	Thickness		Camber	
	x	y	x	y
Leading control points	0	0	0	0
	0	0.5	0.15	0.075
	0.2	0.6	0.3	0.125
	0.45	0.6	0.4	0.125
Trailing control points	0.45	0.6	0.4	0.125
	0.55	0.6	0.5	0.125
	0.8	0.025	0.75	0.075
	1	0	1	0

Table 3.6: Bezier control points - Bez4a4f



a. Fan blade shape



b. Pressure distributions

Figure 3.10: Geometry test: airfoil Bez5a4f.

116 panels. Both meshes were passed to the flow solver to compare the resulting pressure distribution. For fan parameters $\beta_1 = 35°$, $\lambda = 0$, $t/l = 0.900364$, $W_1 = 1.0$, the pressure distributions shown in Fig. 3.10b were calculated. In this case there is excellent correlation between the results. Note that this test demonstrates the ability to have a different number of control points on the leading and trailing curves. Here the Bezier curves on the leading edge have five control points, and the curves on the trailing edge have four control points.

	Thickness		Camber	
	x	y	x	y
Leading control points	0	0	0	0
	0	0.3	0.15	0.08
	0.15	0.35	0.25	0.95
	0.3	0.4	0.3	0.1
	0.45	0.4	0.4	0.1
Trailing control points	0.45	0.4	0.4	0.1
	0.55	0.4	0.5	0.1
	0.8	0.025	0.75	0.05
	1	0	1	0

Table 3.7: Bezier control points - Bez5a4f

3.3 Optimization Techniques

The goal of an optimization problem can be formulated as follows: find the combination of parameters which minimize a given quantity, possibly subject to some restrictions on the allowed parameter ranges. The quantity to be minimized is termed the objective function, $f(\mathbf{x})$; the vector $\mathbf{x}$ contains n parameters, called control variables, which may be changed in the quest for the optimum; the restrictions on allowed parameter values are known as constraints; the allowable space which is to be searched is called the control space or solution space.

For the design optimization problem discussed here, multidimensional optimization algorithms are used that do not require calculation of the gradient of the objective function. The reason for the exclusion of gradient-based techniques is the manner in which constraints are imposed. A penalty function is used to generate a large random objective function value whenever a constraint is violated. The necessity for and nature of the penalty function is further discussed below in section 3.4.3.

Three different multidimensional optimization techniques were compared on a number of design test cases. The downhill simplex method attempts to crawl slowly downhill towards the minimum function value. Simulated annealing employs a random search of the design space and will sometimes accept uphill steps in its attempt to find the global minimum. Differential evolution simulates natural selection, as theorized by Darwin, in its search for the best set of variables.

It is important to understand the distinction between local and global minima. A local minimum of a function is just the lowest point within some given neighbourhood. Aerodynamic objective functions frequently have many local minima. The global minimum of a function is the lowest point that the function will ever reach. In aerodynamic shape optimization, it is quite important to be able to closely approximate the global minimum of the objective function.

3.3.1 Downhill Simplex Method

The downhill simplex method simply follows the local topography until it reaches a minimum. It is generally very fast, but cannot guarantee that a global minimum will be found.

Theoretical Background

A simplex is an N-dimensional figure consisting of $N + 1$ vertices and all the polygonal faces between them. In two dimensions for example, the simplex is just a triangle. After an initial simplex is defined, it is subsequently modified so as to make its way downhill to a minimum where the program is terminated.

The initial simplex is defined with $N+1$ points which must be an initial guess. If one point is labeled the initial starting point $\mathbf{P}_0$ then the other points can be defined using the following equation:

$$\mathbf{P}_i = \mathbf{P}_0 + \mu \mathbf{e}_i \tag{3.28}$$

as shown in Press et al. (1992). The $\mathbf{e}_i$'s are N unit vectors and μ is the characteristic length scale - a guess at how far away from $\mathbf{P}_0$ the actual solution might be.

The simplex is moved downhill through the use of three basic steps: reflection, expansion, and contraction of the simplex. The most common step is the reflection, where the highest point (the one generating the greatest objective function value) is moved through the opposite face of the simplex to a lower point. When possible, the simplex expands in one direction, allowing it to take larger steps. It will also contract around any local minima.

The stopping criteria used is ftol, defined as the fractional convergence tolerance to be achieved in the function value. That is, the decrease in the function value in the final step should be smaller than ftol. Therefore, all new points returned by the algorithm will be within ftol of a local minimum.

Code Development

Two subroutines that implement the downhill simplex method were found in Press et al. (1992). These subroutines are called amoeba.c and amotry.c and were included in the program without modification. The function DownhillSimplex(), coded into the mainline program, sets up the initial simplex, and then calls the subroutine amoeba.c.

3.3.2 Simulated Annealing

As its name implies, simulated annealing exploits an analogy between the annealing process by which a metal cools and freezes into a minimum energy crystalline structure, and the search for a minimum in a more general system. It is considerably more innovative than the downhill simplex method, and has the ability to avoid becoming trapped at local minima.

Theoretical Background

When metals are cooled slowly enough, nature is able to find the minimum energy crystalline structure by redistribution of the atoms as they lose mobility. The Metropolis algorithm (Metropolis, 1958) used by simulated annealing is designed to mimic this process in an optimization routine. The objective function in mathematical optimization is analogous to the energy in a physical system, and the global minimum to the minimum energy state. The implementation of the simulated annealing algorithm requires the following elements: an objective function E that is to be minimized, a control parameter T (analogous to temperature), an annealing schedule for the reduction of T, and a random number generator to provide a random search of the solution space.

The major advantage of the Metropolis algorithm over other methods is the ability to avoid becoming trapped at local minima. The algorithm employs a random search which not only accepts decreases in the objective function $f(\mathbf{x})$, but also some increases. The latter are accepted with a probability $\exp(-\delta f/T)$, where δf is the increase in f, and T is given by the annealing schedule. The annealing schedule thus determines the degree of uphill

movement permitted during the search and is critical to the algorithm's performance.

The principle underlying the choice of a suitable annealing schedule is easily stated. The initial temperature should be high enough to melt the system completely and should be reduced towards its freezing point as the search progresses. Many possibilities exist, however, and choosing an annealing schedule for practical purposes is something of a black art (Bounds, 1987). The annealing schedule used here is that suggested by Vanderbilt and Louie (1984). An initial temperature $T = T_0$ is given, and after every m steps T is multiplied by ε, a factor between zero and one. Both ε and m must be chosen by experiment. The best annealing schedule is problem dependent, typically requiring many experiments to be found.

In the simulated annealing version given by Press et al. (1992) and used here, the system state is described by a simplex of $N + 1$ points, as in the downhill simplex method. The simplex moves are the same as those described in section 3.3.1 above. At any temperature T the simplex scales itself to the approximate size of the region that can be searched at that temperature. Then, using the Metropolis algorithm, the control space is freely examined among local minima of depth less than about T. As the annealing schedule lowers T, the simplex shrinks, and the number of such minima qualifying for frequent visits is gradually reduced. If the temperature is reduced sufficiently slowly, it is likely that the simplex will converge to the global minimum.

Code Development

Two subroutines that implement simulated annealing were found in Press et al. (1992). These subroutines are called amebsa.c and amotsa.c and were included in the pro-

gram without modification. The function SimulatedAnnealing(), coded into the mainline program, sets up the initial simplex, then calls the optimizing function, amebsa.c. This subroutine is called from within a for loop which decreases the temperature T according to the annealing schedule after each function call to amebsa.c.

3.3.3 Differential Evolution

Genetic algorithms interpret the value of a function at a point as a measure of that point's fitness as an optimum. Then, guided by the principle of survival of the fittest, an initial population of vectors is transformed into a solution vector through repeated cycles of mutation, recombination, and selection. Differential evolution is a recently developed genetic algorithm which is proving to be quite effective. An accessible description of it is given in Price (1997).

Theoretical Background

The overall structure of the algorithm resembles that of most other population-based searches. Two arrays are maintained, each of which holds a population of NP, D-dimensional, real-valued vectors. The primary array holds the current population while the secondary array accumulates vectors that are selected for the next generation. Selection occurs by competition between the existing vectors and trial vectors. The trial vectors are formed through mutation and crossover (recombination) of the vectors in the primary array. In all, just three parameters control evolution: the population size, NP; the weight applied to the differential in mutation, F; and the constant that mediates the crossover operation, CR.

Mutation is an operation that makes small random alterations to one or more parameters of an existing population vector. Mutation is crucial for maintaining diversity in a population, and is typically performed by perturbation. That is, a vector, $\mathbf{X}$, is perturbed by adding to it a differential vector, $\mathbf{dX}$. A convenient source of appropriately scaled perturbations $\mathbf{dX}$ is the population itself. Every pair of vectors, $(\mathbf{X}_a, \mathbf{X}_b)$, defines a vector differential, $\mathbf{X}_a - \mathbf{X}_b$. When these two vectors are chosen randomly, their weighted difference can be used to perturb another vector, $\mathbf{X}_c$:

$$\mathbf{X}'_c = \mathbf{X}_c + F(\mathbf{X}_a - \mathbf{X}_b). \tag{3.29}$$

The weight, F, is a user-supplied constant in the range $0 < F \leq 1.2$. The optimal value of F for most functions lies in the range $0.4 \leq F \leq 1.0$.

Recombination (or crossover) provides an alternative and complementary means of creating viable vectors. Designed to resemble the natural process by which a child inherits DNA from its parents, recombination builds new parameter combinations from the components of existing vectors. Differential evolution uses a nonuniform crossover that can take child vector parameters from one parent more often than it does from the other. By using components of existing population members to construct trial vectors, recombination efficiently shuffles information about successful combinations, enabling the search for an optimum to focus on the most promising areas of the solution space.

Once in every generation, primary array vectors are targeted for recombination with a vector like $\mathbf{X}'_c$ (3.29) to produce a trial vector, $\mathbf{X}_t$. Thus the trial vector is the child of two parents: a noisy random vector and the primary array vector against which it must compete. Which parent contributes which trial vector parameter is determined

by a series of binomial or exponential experiments, mediated by the crossover constant, CR, where $0 \leq CR \leq 1$. Binomial crossover starts at a randomly selected parameter, and then the source of each trial vector parameter is determined by comparing CR to a uniformly distributed random number from within the interval $[0, 1)$. If the random number is greater than CR, the trial vector gets its parameter from the target vector in the primary array. Otherwise, the parameter comes from the noisy random vector, $\mathbf{X}'_c$. In exponential crossover, trial vector parameters are chosen from $\mathbf{X}'_c$ until the random number generator produces a value larger than CR (or until all parameters have been determined). The remaining parameters then come from the primary array vector. In both methods, when $CR = 1$, every trial vector parameter comes from $\mathbf{X}'_c$, making the trial vector an exact replica of the noisy random vector.

Once new trial solutions have been generated, selection determines which among them will survive into the next generation. Each child is pitted against one of its parents, which one being chosen at random. Only the fitter of the two is then allowed to advance into the next generation.

Choosing Differential Evolution's control parameters, NP, F, and CR is seldom difficult, and some general guidelines can be given. Ordinarily, NP ought to be about five to ten times D, the number of parameters in each vector. A good initial attempt for F is 0.5. If the population converges prematurely, F and/or NP should be increased. In general, CR should be as large as possible without causing the population to become devoid of diversity. Consequently, try $CR = 1.0$ or $CR = 0.9$ to see if a quick solution is possible before trying $CR = 0.1$. In fact, the range in which the control parameters are effective is

usually quite large.

Code Development

The C functions that implement Differential Evolution, version 3.6, were downloaded from the author's webpage: http://www.icsi.berkeley.edu/~storn/code.html. A few minor modifications were required for the primary function, de36(), to be properly integrated into the design optimization algorithm. Function parameters were added to enable information such as the objective function pointer, the final optimized control variables, and the final error value to be passed between the modules.

Also, the default definition of the initial population had to be modified. Originally, de36() was coded to choose the population of the first generation completely at random. With all the geometric constraints imposed in the design problem, however, the random search for vectors in the solution space was rather unsuccessful. The initialization was thus recoded to choose randomly vector elements from within a simplex surrounding the initial fan blade shape. The initial simplex was then designed to ensure the first generation consisted only of viable shapes.

A variety of differential evolution strategy options are given in the code. The best results were obtained using strategy 3 with $NP = 200$, $F = 0.85$, and $CR = 1$. This strategy uses exponential crossover. The vector $\mathbf{X}_c$ chosen for mutation is a combination of two vectors: the best one found so far and one randomly chosen from the population.

The function DifferentialEvolution() was coded into the mainline. It reads the file containing the differential evolution parameters (discussed below in section 3.4.1), and then calls de36().

3.3.4 Testing of Minimization Algorithms

The code for each optimization algorithm was retrieved from a reliable source. The downhill simplex and simulated annealing routines were copied from a Numerical Recipes diskette (Press et al., 1992), and differential evolution was downloaded from the author's webpage. The effectiveness of each routine has already been tested and verified by these authors. If any modification was made, it was to the interface and not to the optimization algorithm itself. Thus the only testing required was to run design cases on the integrated aerodynamic optimization code using the three different minimizers. Some simple tests are described in section 3.4.5. Additional test cases were used to compare the algorithms (section 4.1).

3.4 Program Integration

The pieces are now in place to construct the fan blade shape optimization program. The main function driver integrates the flow solver, shape definition, and optimization routines into a program that will produce a fan blade designed to perform according to some prescribed design criteria. The design criteria, for this problem, will consist of a pressure distribution, and the inlet and outlet angles of the flow. A flow chart indicating the overall structure of the code is given in Fig. 3.11.

3.4.1 Preprocessing

A large number of input files must be read before starting the optimization process. These files describe most of the problem parameters, including: target pressure distribution;

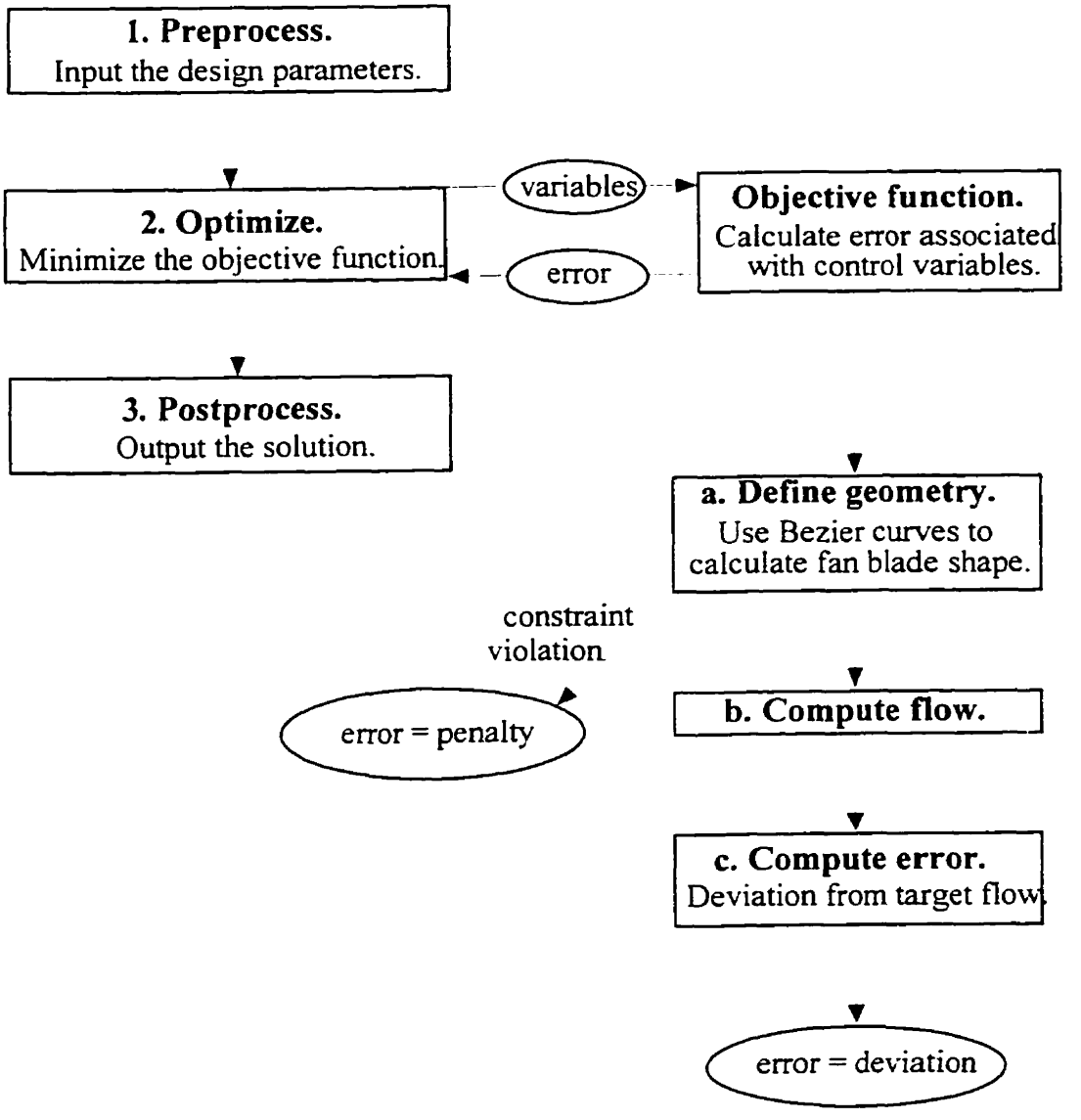


Figure 3.11: Flow chart for integration of all components of the design algorithm.

fan performance parameters; initial fan blade shape; initial stagger angle; optimization routine to be used; optimization routine parameters. In fact, the only problem parameters which must be hard-coded are the number of control points on the leading and trailing Bezier curves. After the problem parameters are determined, the preprocessing is complete, and the appropriate optimizing routine is called.

The use of input files enabled multiple tests to be performed efficiently. Had the parameters been hard-coded as global constants, the code would have needed recompiling for each run. Also, while a more interactive program is sometimes desirable, this would have required the re-typing of much input data. The code could, however, easily be modified to accept parameter input interactively from a terminal.

Seven resource files are used: Target.r, InitCamb.r, InitThic.r, Design.r, Optimizr.r, SAin.r, and DEin.r. Most are used to input the file name containing specific test parameters. This allows different test files to be stored and reused without being renamed. Each of these files lists a string and an integer, on separate lines.

Target.r lists the filename which contains the target pressure distribution, and the number of points in that distribution. The pressure distribution file itself is comprised of two columns of numbers. The first column holds the x-values and the second holds the y-values. It must contain an even number of data points.

InitCamb.r and InitThic.r give the filenames containing the initial Bezier control variables for the camber and thickness curves, respectively. The integer value is either 1 or 0, and determines whether or not, respectively, the curve will be optimized. This allows the user to optimize only one, or both of the thickness and camber profiles. It should be

	Thickness		Camber	
	x	y	x	y
Leading control points	0	0	0	0
	0	0.5	0.15	0.075
	0.2	0.6	0.3	0.125
	0.45	0.6	0.4	0.125
Trailing control points	0.45	0.6	0.4	0.125
	0.55	0.6	0.5	0.125
	0.8	0.025	0.75	0.075
	1	0	1	0

Table 3.8: Initial Bezier control points.

emphasized that the files named here do not contain all of the control points for the curve. Only the variables are listed, and they must be listed in the order in which the Geometry routine reads them. This order is specified in section 3.2.2.

The initial Bezier control point variables used for most design cases were: camber control variables = {0.15, 0.1, 0.3, 0.15, 0.45, 0.6, 0.78, 0.05}, thickness control variables = {0.03, 0.2, 0.08, 0.4, 0.6, 0.8, 0.02}. The resultant camber and thickness control points are listed in Table 3.8. This initial curve design enabled the addition of $\mu \in (-0.02, 0.07)$ to any given control variable without violating any of the constraints listed in section 3.2.3.

Design.r names the file describing the design fan performance parameters. The integer is a dummy in this case. The actual file containing the desired performance parameters will consist of five numerical values in the following order: initial pitch/chord ratio, initial stagger angle, inlet velocity, inlet angle, and outlet angle.

Optimizr.r lists the optimizer to be used and the integer N where the desired optimal tolerance is $tol = 1/N$. The string characterizing the optimizer must be one of the following: "DE", "DownSimp", or "SimAnneal".

The last two input files are required to input optimization parameters for the simulated annealing and differential evolution routines. The file SAin.r names the file containing the annealing schedule parameters, and a dummy integer. Annealing parameters must be listed there in the following order: the number of iterations m, the temperature reduction ε, and the initial temperature, T_0. DEin.r gives the filename that lists the differential evolution parameters, and a dummy integer. Differential evolution parameters must be listed in the following order: the strategy number, the maximum number of generations, the output refresh cycle, the number of parameters D, the population size NP, the upper parameter bound for the initial variables, the lower parameter bound for the initial variables, the weight factor F, the crossover factor CR, and an integer seed for the random number generator.

3.4.2 Control Variables

The control variables are the variables which are to be optimized. That is, they are the variables of the objective function. The control variables are comprised principally of the parameters used to define the Bezier curve control points. For some design problems, the list of control variables may also include the stagger angle and the pitch/chord ratio of the fan blades.

The Bezier control variables are described in section 3.2.2. These parameters will define the thickness and camber curves, and thus the geometric shape, of any fan blade tested. They are constrained to be in the range $[0, 1]$, as discussed in section 3.2.3. The user may optimize both camber and thickness curves, but has the option of optimizing only one of these, keeping the other fixed.

The performance of a fan also depends to some degree on the manner in which the fan blades are staggered. One problem which may be defined is that of designing a fan blade to generate a flow through the fan characterized by given inlet and outlet angles. In this case, the stagger angle is one of the parameters to be optimized.

The stagger angle characterizing a compressor fan is typically in the range $(0°, 90°)$. It is input to the flow solver in degrees, and then converted into radians. The optimization routines are most efficient if the variables all have roughly the same range, so a simple linear transformation is required to map the stagger angle parameter onto $[0, 1]$, the range of the Bezier control variables. That is, the stagger angle λ is transformed to the nth design parameter, x_n, using the formula

$$x_n = \frac{\lambda - a}{b - a}, \tag{3.30}$$

where $a = 0°$, $b = 90°$. Then, before being passed to the flow solver, the inverse transformation is used to recover the actual stagger angle. It should be noted that the optimization routines will search outside the given range of $[0, 1]$ in case the optimal λ lies outside the range $[a, b]$.

Further tests suggested the existence of an optimal pitch/chord ratio, t/l. The pitch, t, of a fan measures the distance between successive fan blades in the cascade. The chord l is the length of any given fan blade. The pitch/chord ratio, then, is a non-dimensional measure of the spacing of the blades in a fan. This was added to the list of control variables in much the same way as the stagger angle, but is constrained to be positive.

When four control points are used to define each of the leading and trailing edge

curves, fifteen variables are required to determine all control points for the Bezier curves, eight for the camber curve and seven for the thickness curve. Together with the stagger angle and pitch/chord ratio, then, seventeen control variables are used in this case if optimizing both curves. Adding a control point to either the leading or trailing edge curves requires the addition of four new control variables. For example, using five control points on the leading edge and four on the trailing edge makes the optimization problem 21-dimensional.

3.4.3 The Objective Function

The objective function is the function which is to be minimized. Its pointer is passed to the optimizing routine. The objective function must calculate, in some sense, the error between the design requirements and the performance of a fan determined by a test set of control variables. As such, this error function is the component that integrates the shape definition, flow solver, and optimization routines.

Penalty Function

The objective function is coded in the routine Error(). Before describing its structure, however, it is necessary to discuss the use of a penalty function. Initially, the Geometry() function was coded to modify any airfoil shape that violated the constraints listed in section 3.2.3. Upon integration and testing, however, this technique was found to generate many flow calculations of very similar airfoils. That is, the modifications made to a large set of constraint-violating airfoils generated a small set of similarly shaped fan blades. In most tests, the constraint violations encompassed 20% - 50% of the total number of flow calculations. This approach was thus deemed an inefficient use of the computing

resources.

An alternate approach to shape modification is the use of a penalty function. When a constraint is violated, the penalty function simply assigns a large random number to the objective function. The optimizer thus rejects variables that violate the constraints, and focuses the search on the subset of the design space in which constraint violations do not occur.

To accomodate this, the Geometry() routine was coded to return a value of FALSE any time a constraint is violated. This is done prior to any Bezier curve calculations. When the error function reads a return value of FALSE from Geometry(), Error() returns a random number in the range $[5000, 10000]$, without bothering to calculate the flow. In this way, unnecessary costly flow calculations are replaced by the simple generation of a large random number, which is then rejected by the optimizing routine.

Overall Structure

The first step of the objective function Error() is to compute the flow. It first transforms the stagger angle variable, x_n, back into the correct range, using equation (3.30). The control point variables are then copied into two arrays, one for each of the camber and thickness curves, keeping in mind that both curves are not necessarily being modified by the optimization process. These Bezier control point variables are passed to the Geometry() function. If any constraints are violated, Geometry() returns FALSE, upon which Error() immediately returns the penalty - a large random number. Otherwise, Geometry() calculates the data points defining the shape of the airfoil. These data points are passed to the FlowSolver() function, which calculates the pressure distribution and outlet angle.

After computing the flow, Error() must compare the pressure distribution and outlet angle of the test airfoil with those of the target. The x-values of the airfoil data points computed by Geometry() will not necessarily correspond with the x-values of the target pressure distribution. Thus quadratic interpolation is used to approximate the pressure coefficients of the test fan blade at the x-values of the target distribution. This interpolation makes use of the routine polint.c, found in Press et al. (1992). Note that the pressure distribution is a closed curve. Thus special care was taken to ensure that the pressure coefficients compared corresponded to the same surface, upper or lower, of the fan blade.

The L_2-error norm is then used as a measure of the deviation between the design requirements and the performance of the test fan blade. Thus the error associated only with the test and target pressure distributions is

$$\|e\|_{L_2} = \sqrt{\sum_{i=1}^{T}(\sigma_i - \tau_i)^2}, \tag{3.31}$$

where T is the number of points in the target pressure distribution, $(\sigma_i)_{i=1}^{T}$ is the array of pressure coefficients interpolated from that of the test fan blade, and $(\tau_i)_{i=1}^{T}$ is the target pressure distribution. To include the outlet angle in the error, it becomes

$$\|e\|_{L_2} = \sqrt{(\beta_{2_s} - \beta_{2_t})^2 + \sum_{i=1}^{T}(\sigma_i - \tau_i)^2}, \tag{3.32}$$

where σ, τ, and T are as above, β_{2_s} is the outlet angle of the test fan, and β_{2_t} is the target outlet angle.

3.4.4 PostProcessing

Upon completion, an optimization routine passes relevant data back to the main function, including the optimized set of variables, the error value, and the number of function calls. This data is then formatted and output to the user, both at the terminal and into files.

The most important output information is the fan blade shape and its associated pressure distribution. In order to return this information to the user, the boolean variable Optimized is set to TRUE and passed back to the flow calculation routine along with the optimized set of variables. The geometry routine then outputs the Bezier control points and fan blade data points to files. The flow solver outputs the resultant pressure distribution. These data files can then be imported and plotted by software such as Mathcad or Maple.

Information about the optimization process is output to the screen. The L_2-error norm is given (that is, the final value of the objective function). The total number of flow calculations and constraint failures are also listed. Resultant flow information is shown, including the inlet and outlet angles, stagger angle, pitch/chord ratio, and lift coefficient. The user is then given the option to calculate streamlines for the designed blade. Finally, information about the output data files is given, including the file names and number of data points contained in each.

3.4.5 Testing and Verification

Two types of tests were performed. First, the interpolation used in the error conmputation was verified. The several simple design tests were run to ensure proper

integration of all modules in the code.

The quadratic interpolation required above to compare the pressure distributions was tested on a number of the test cases described above in section 3.1.3. Fig. 3.12 compares the pressure distribution calculated around airfoil C4/70/C50. $\beta_1 = -35°$, with the distribution interpolated from it at different x-values. This is just one example of the excellent agreement between the original and interpolated pressure distributions.

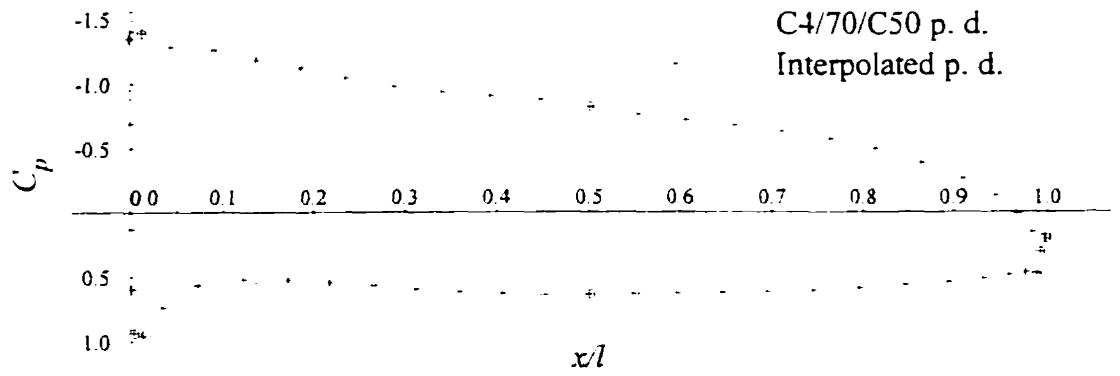


Figure 3.12: Pressure distribution interpolation test.

The error function (3.32) was also verified on numerous cases. The interpolated distribution from a test shape and the target distribution were both input into Mathcad. The error function was then coded directly into Mathcad to confirm the L_2-error norm calculated by the code.

The remaining tests were used to verify the proper integration of all sections of the code. The stagger angle and pitch/chord ratios were held fixed. The tests were performed using inverse design problems for which the solution was already known. Given a blade shape, the vorticity method flow solver was used to compute the pressure distribution

around the blade for a given set of design criteria. Then that distribution was used as the target of an inverse design problem. The solution should generate the shape which is in fact known to be optimal. To challenge the code, the data points used on the original blade were different from those that the code would design.

As a simple first test, fan blade shape Bez5a4f (Fig. 3.10) is used both as the target and as the initial blade shape. Each of the three optimization algorithms were able to converge quickly to the target, with an L_2-error norm of less than 0.1. Each of the optimization routines was further tested on four other design cases in section 4.1. The initial shape for these tests is that determined by the control points given in Table 3.8. The reasonably good performance of all algorithms on at least the less challenging designs was proof enough that the integrated code was performing as expected.

Chapter 4

Results

The inverse design optimization algorithm described in Chapter 3 is applied to two types of problems. First, the performances of the three optimizers discussed above are compared. Several test cases will be used to determine the "optimal" routine. Second, using the best optimizer, the algorithm is applied to an inverse design problem. A Liebeck pressure distribution is chosen as the target, and several fan blade shapes are designed.

4.1 Optimal Optimization in Aerodynamic Design

When different optimization algorithms are used to solve the same minimization problem, the results will differ. It is of interest, then, to compare and evaluate the performance of the minimizing routines described in section 3.3. In this section, each of the optimization algorithms is run on four aerodynamic design test cases. Differential evolution will be shown to be the best optimization strategy of those discussed in this work.

4.1.1 Problem Definition

It is important, first, to discuss in what sense one optimization algorithm might be considered better than another. There are two considerations, efficiency and effectiveness. An efficient algorithm can arrive at a near-optimal objective using a minimum number of function evaluations. An effective algorithm, on the other hand, will consistently find a function value very close to the global minimum, and will not become trapped by a local minimum. While both are important, clearly effectiveness takes precedence over efficiency when designing aerodynamic shapes. It would be practically pointless to publish an airfoil shape which was obtained quickly, but does not meet the design requirements.

In this evaluation of the optimization algorithms, a greater weight will be placed on the algorithm's ability to search out the lowest possible error. However, consideration will be given to the cost required to arrive at a result. Measuring this efficiency is made somewhat ambiguous by the use of the penalty function, though. Technically, a constraint violation does run through the objective error function, but the computational cost is insignificant compared to that of a flow solution. Thus the number of function evaluations (NFE's) is defined to be the number of flow calculations performed by the code. This best reflects the actual computational expense of the algorithm.

Four different inverse design test cases were used to test the algorithms. In each case, the actual airfoil shape which generates the given target pressure distribution is known. This allows comparison not only between the designed and target flows, but also between the designed shape and the true optimum. Because the target distribution is the resultant flow of a known shape, the pitch/chord ratio and stagger angle of the fan, and the inlet

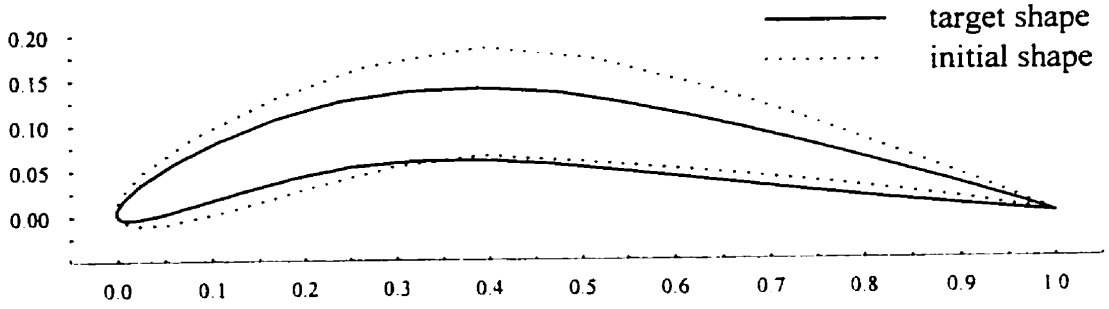
velocity, inlet angle, and outlet angle of the flow are also known.

For the purpose of comparing the algorithms then, these latter fan parameters were not part of the optimization scheme. That is, the stagger angle, pitch/chord ratio, inlet velocity, and inlet angle were held constant, and the outlet angle was not part of the objective function. The control variables thus consisted only of the parameters necessary to define the airfoil shapes. Similarly, the objective function simply measures the deviation between the required pressure distribution and that of the fan blade defined by the function's variables (equation (3.31)).

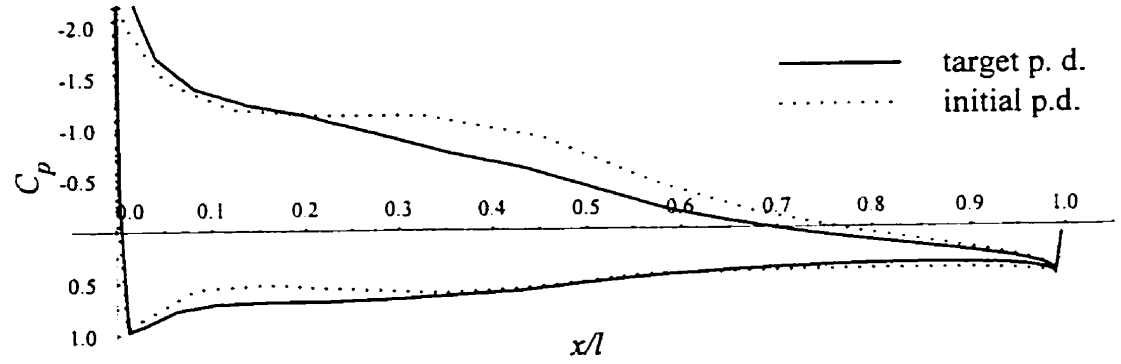
All Bezier curves which are used to define the camber and thickness curves are parametrized from four control points. This can be done with fifteen control variables, so the optimization problem is fifteen-dimensional. The maximum element length used for all cases is Lmax = 0.05. This will generate meshes which might be too coarse for an actual design problem, but which adequately demonstrate the properties of the optimizers.

4.1.2 Case 1: Bez5a4f

The first shape is the Bezier-generated fan blade Bez5a4f. This was used in section 3.2.4 to test the shape definition module. The 60 data points were input into the flow solver, with fan parameters $\beta_1 = 35°$, $\lambda = 0$, $t/l = 0.900364$, $W_1 = 1.0$. The resulting pressure distribution was used as the target of the inverse design problem. The original shape was defined using five Bezier control points on the leading edge curves, but only four are used in this optimization process. The initial curve used for this test case is Bez4a4f, another Bezier-generated fan blade used in section 3.2.4. The initial shape and pressure distribution are actually very similar to those of the target fan blade (Fig. 4.1).



a. Fan blade shapes



b. Pressure distributions

Figure 4.1: Initial and target curves: Case 1.

m:	15	20	25	30	30	30	30	30	30	35	40	40
ϵ:	.5	.55	.55	.6	.57	.55	.54	.52	.5	.55	.55	.5
Error:	2.8	3.3	.40	1.5	1.4	.38	.90	.91	1.3	3.2	1.2	.98

Table 4.1: Comparison of annealing schedule parameters.

Optimizer	**Error**	**NFE's**
Downhill Simplex	0.095	1 700
Simulated Annealing	0.376	2 600
Differential Evolution	0.096	47 000

Table 4.2: Comparison of optimization algorithms: Case 1.

The downhill simplex and differential evolution optimizers were stopped after reaching an error value less than 0.1. Downhill simplex attained this value after only 1700 function evaluations, needing about fifteen minutes to run on an (already obsolete) 486 DX-100 PC. Differential evolution required a comparatively massive 47 000 test flow calculations, taking five hours to complete. This run time would be improved considerably if using a more recently designed computer.

Before simulated annealing was able to arrive at a reasonable solution, an effective annealing schedule had to be found. This involved experimenting with the parameters m and ϵ, as discussed in section 3.3.2. Bender (1996) suggests that m should be roughly twice as large as the number of control variables, and found $\epsilon = 0.5$ to be the most effective. Table 4.1 shows a range of experimental parameters and the associated errors. The best objective function value, an error of 0.376, was obtained with $m = 30$ and $\epsilon = 0.55$. These were used for the remainder of the test cases. With additional experimentation, better annealing schedule parameters might be found, but this is effectively another optimization problem, and an impractical one at that.

All three optimizers performed quite well on this test case. Their respective errors and number of function evaluations are given in Table 4.2 for comparison. The target and designed blades are depicted in Fig. 4.2. The downhill simplex and differential evolution methods were able to find a solution with an extremely low associated error. Differential evolution is much less efficient, however, requiring over 25 times the number of function evaluations. Although simulated annealing did find a shape very close to the target, the search for the best annealing schedule was rather time-consuming.

4.1.3 Case 2: C4/70/C50 +35° Inlet Angle

The second target shape is the C4/70/C50 profile, used in section 3.1.3 to test the flow solver module. The 50 data points were input into the flow solver, with fan parameters $\beta_1 = 35°$, $\lambda = 0$, $t/l = 0.900364$, $W_1 = 1.0$. The resulting pressure distribution was used as the target of the inverse design problem. The initial shape used for this and the remaining test cases is that described in section 3.4.1. The initial shape and pressure distribution are less similar to the target than in the first case (Fig. 4.3).

None of the optimizers were able to converge to solutions under 0.1 in this case. The downhill simplex run was stopped after 1200 function calls when the simplex converged to a tolerance of 10^{-5} with an objective function value of 0.396. The pressure distributions and blade shapes are reasonably similar (Fig. 4.4). The two population-based algorithms were stopped when all population members were within a tolerance of 10^{-5}. Simulated annealing took 1700 function calls and converged to an error of 0.559. Fig. 4.5 shows a close resemblence between the design and target shapes. Differential evolution required 57 000 flow calculations, but converged to an error of 0.195. The agreement with the

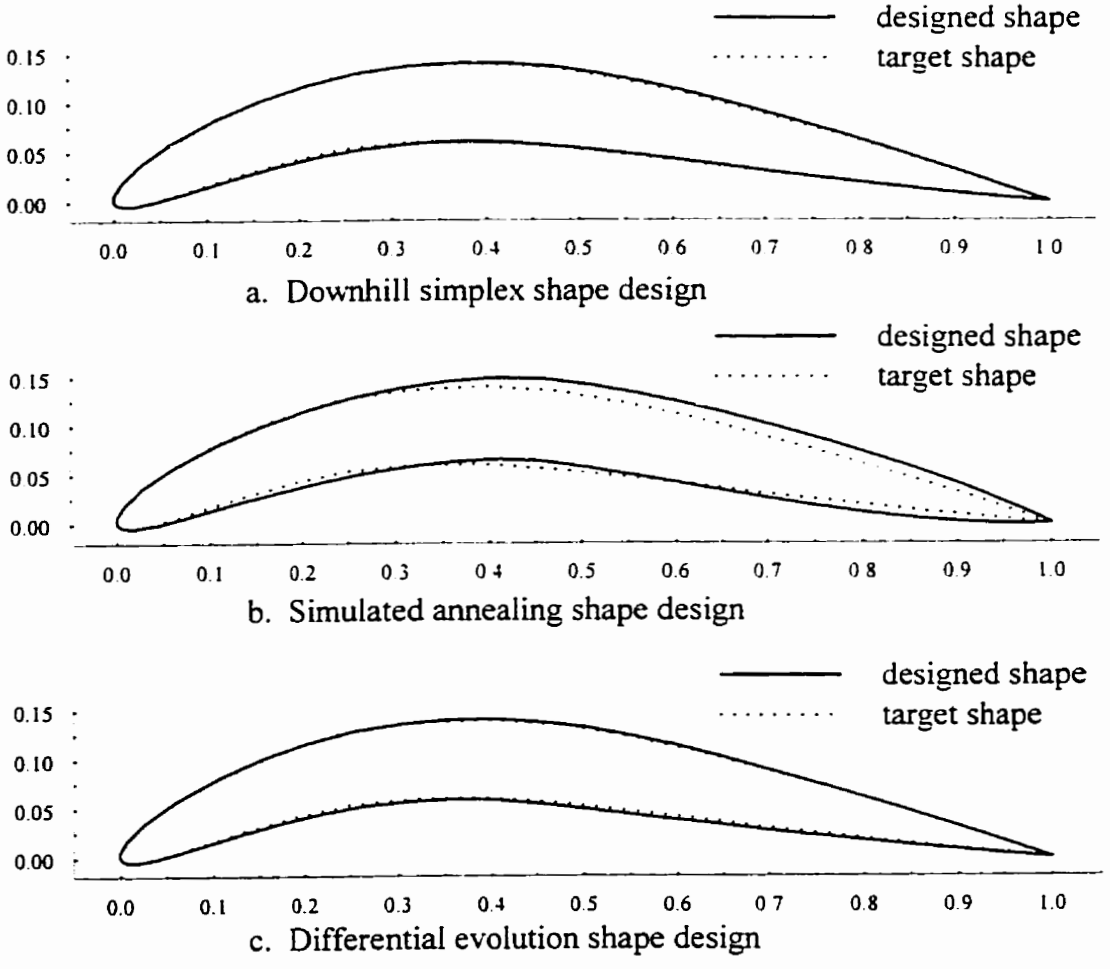


Figure 4.2: Comparison of designed and target blades: Case 1.

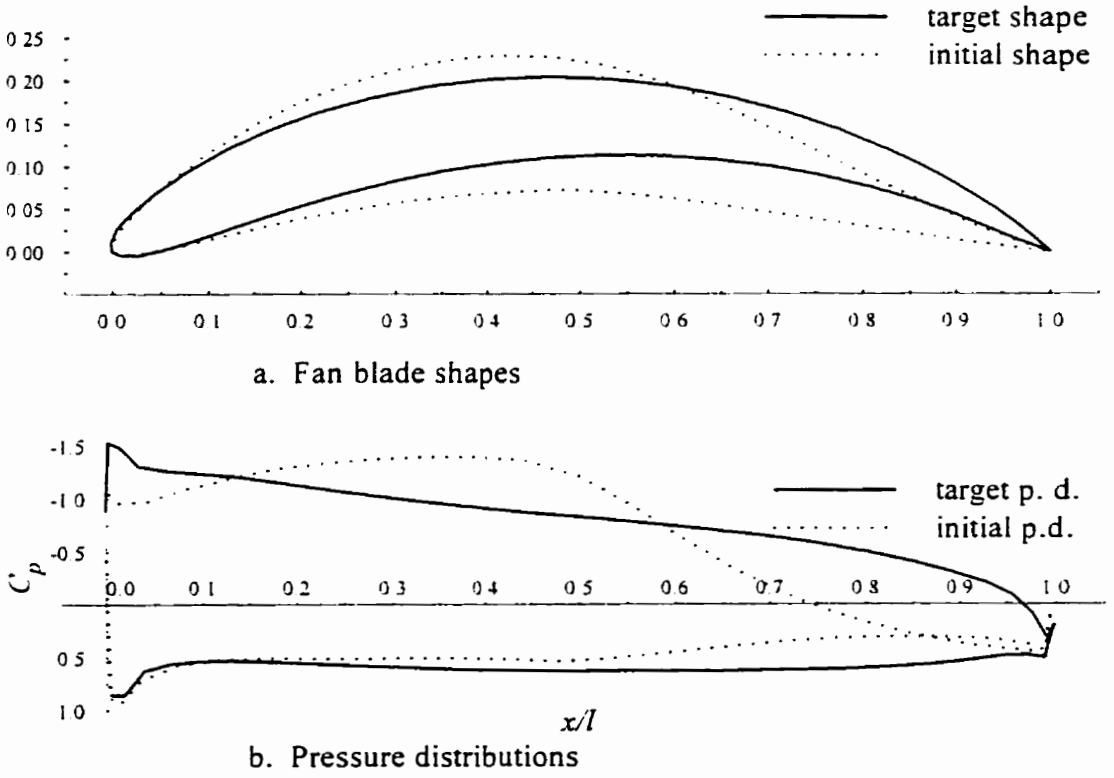


Figure 4.3: Initial and target curves: Case 2.

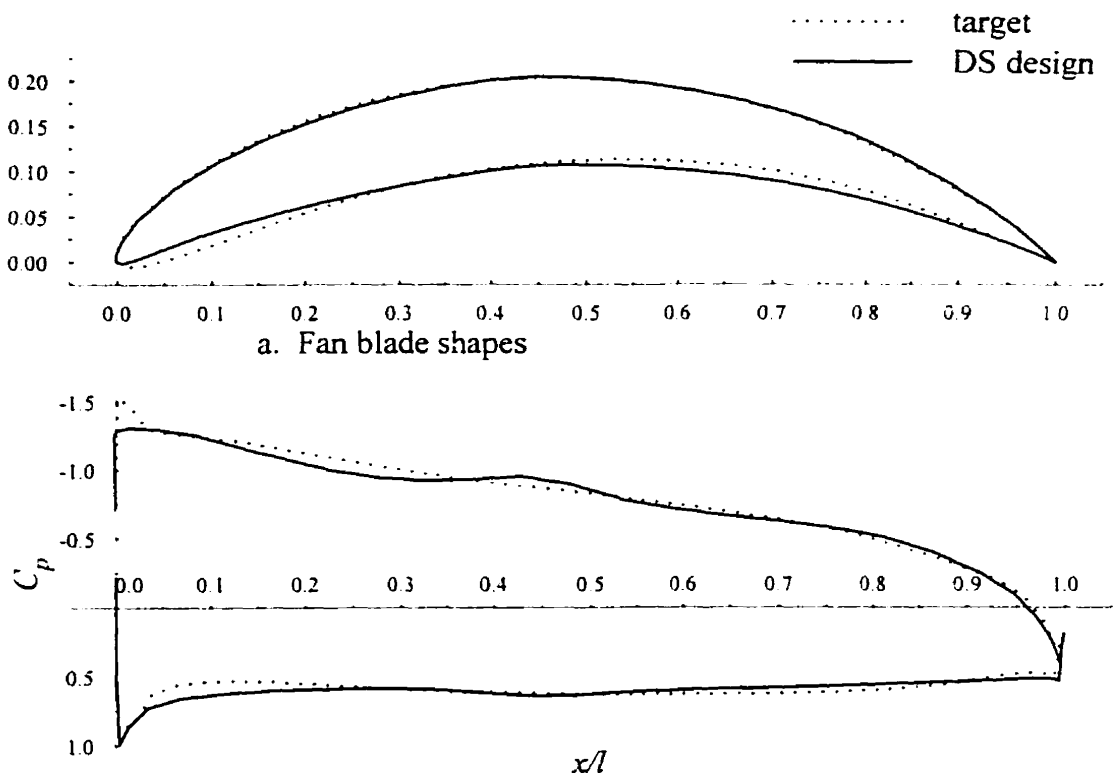
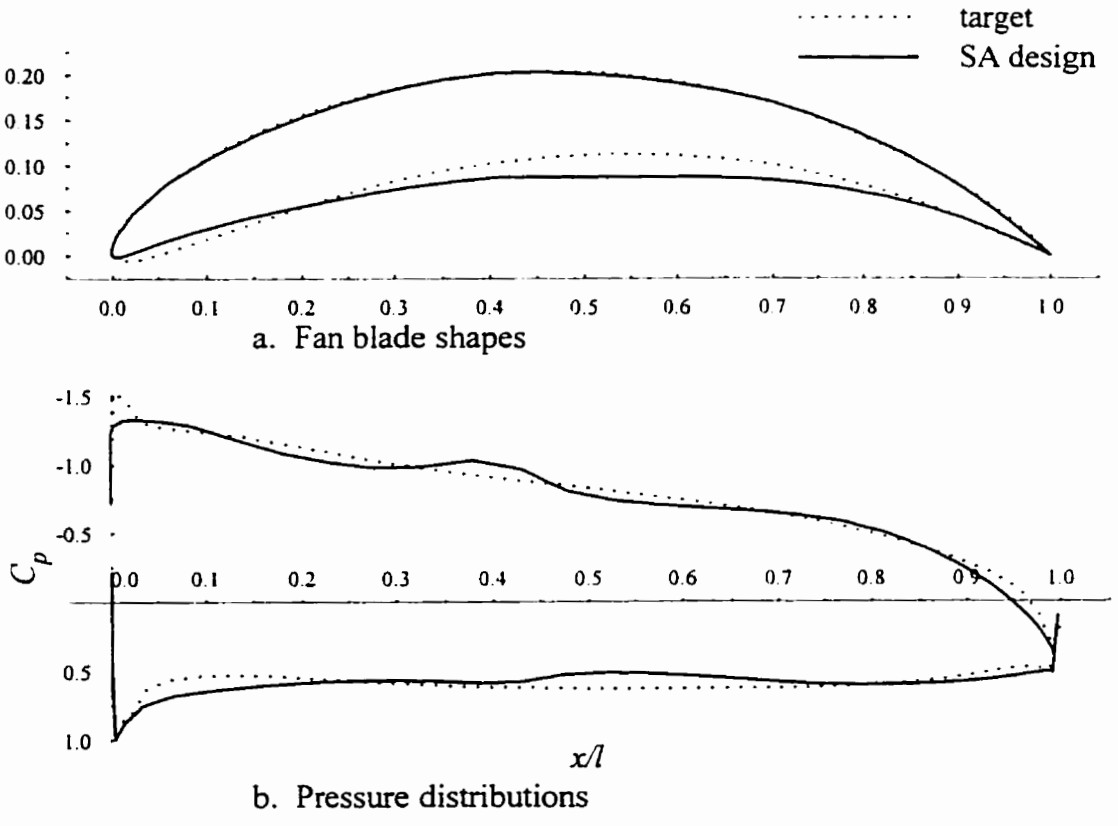


Figure 4.4: Downhill simplex design: Case 2.

target shapes, compared in Fig. 4.6, is clearly superior to the results from the other two optimizers.

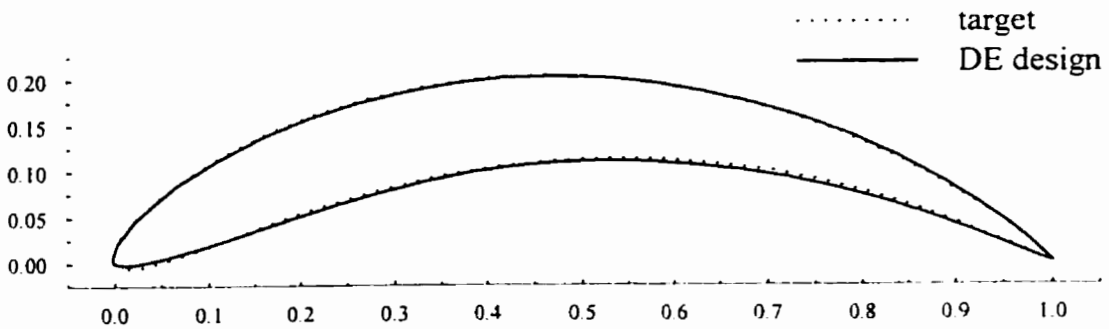
Again, each optimizer was able to closely approximate the target shape. Table 4.3 summarizes the respective errors and costs. This test was slightly more demanding than the first. Objectively, the differential evolution solution was only twice as good as that of the downhill simplex method, its nearest competitor. Subjectively, however, the differential evolution pressure distribution is much more similar to the target than is the case for the other algorithms. It was still much more costly, this time by a factor of almost



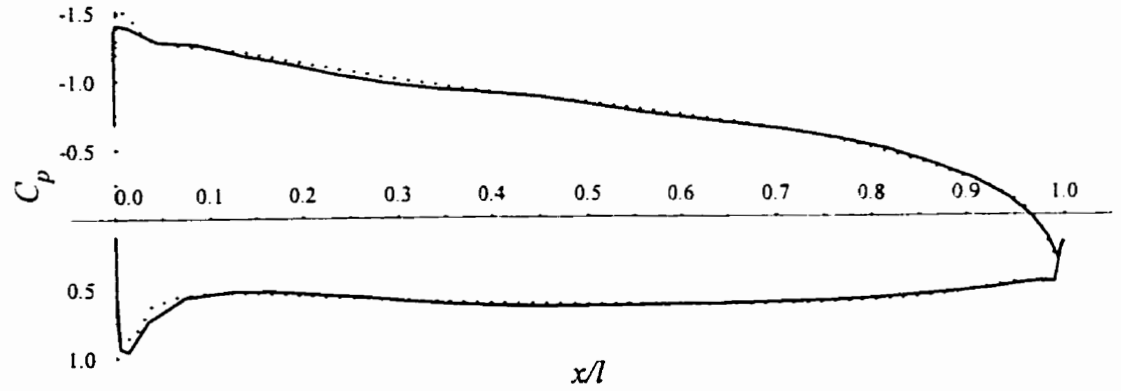
a. Fan blade shapes

b. Pressure distributions

Figure 4.5: Simulated annealing design: Case 2.



a. Fan blade shapes



b. Pressure distributions

Figure 4.6: Differential evolution design: Case 2.

Optimizer	Error	NFE's
Downhill Simplex	0.396	1 200
Simulated Annealing	0.559	1 700
Differential Evolution	0.195	57 000

Table 4.3: Comparison of optimization algorithms: Case 2.

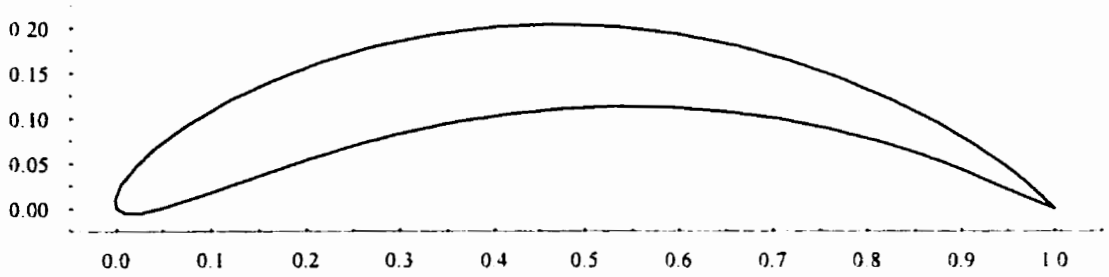
50.

These first two design cases used an initial geometry which was very similar to the target, and produced a very similar pressure distribution. This has stacked the odds in favour of the downhill simplex method, since it is very good at finding local minima close to and within the region of the initial simplex. The next two cases provide a more rigorous test of the optimization routines.

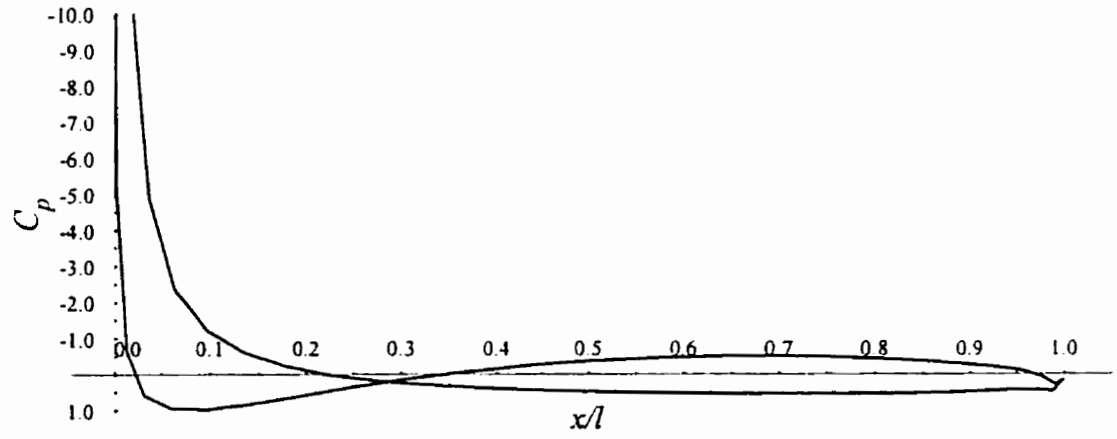
4.1.4 Case 3: C4/70/C50 -35° Inlet Angle

The same blade shape is used as in the second case, but the inlet angle was changed to $\beta_1 = -35°$. The other parameters remained constant: $\lambda = 0$, $t/l = 0.900364$, $W_1 = 1.0$. The resultant pressure distribution, used as the target for optimization, is shown in Fig. 4.7.

Again a tolerance of 10^{-5} was used to stop each run. Both downhill simplex (Fig. 4.8) and simulated annealing (Fig. 4.9) were completely incapable of finding the target blade. Downhill simplex stopped computing significantly different figures after only 960 iterations, and simulated annealing after 2800 population members had been tried. Neither method was able to achieve errors smaller than 6. Differential evolution, on the other hand, converged to an error of 3.36 after 57 000 function evaluations. More significantly, a visual



a. Fan blade shape



b. Pressure distribution

Figure 4.7: Target curves: Case 3.

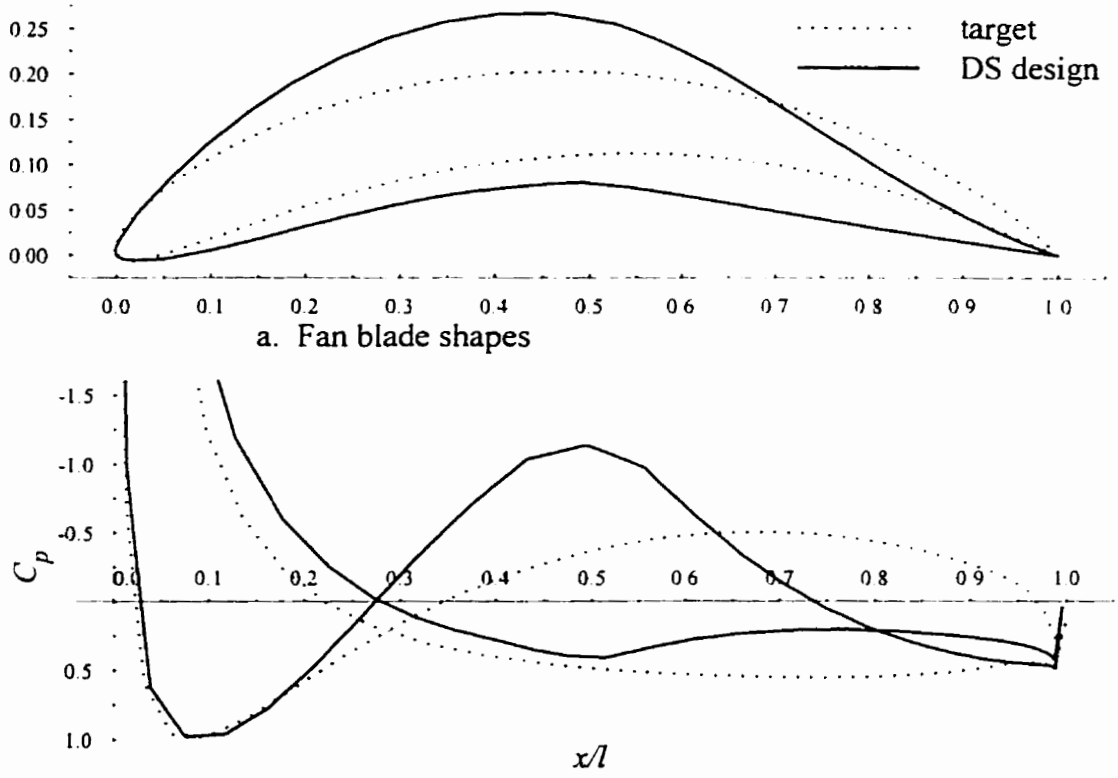


Figure 4.8: Downhill simplex design: Case 3.

inspection of the results in Fig. 4.10 demonstrates the close similarity of the differential evolution design with the target.

The pressure distribution was much more difficult to match in this case. The primary obstruction to convergence occurs in the first 10% chord of the target. This negative pressure region near the nose of the fan blade is of very high magnitude. Any deviation in this region produces a much larger error than when the pressure magnitude is smaller. The summary of the optimizers' performances in Table 4.4 doesn't really give differential evolution the credit it is due. It was remarkably effective in matching the pressure distribution

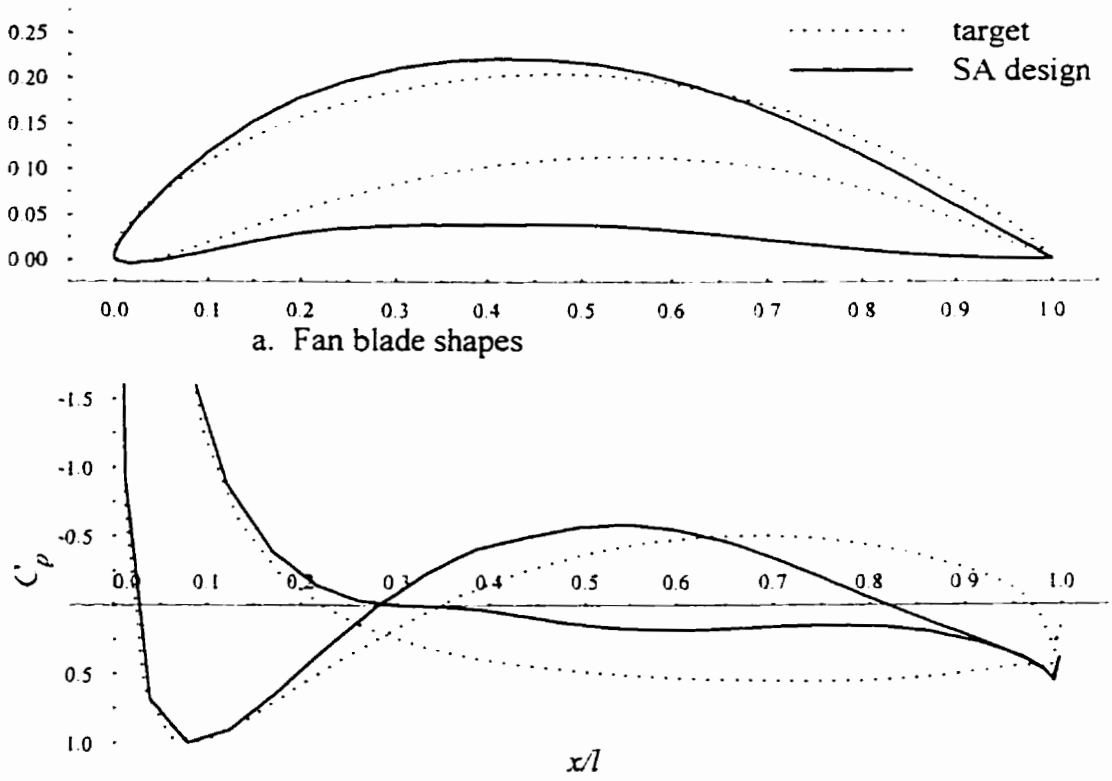


Figure 4.9: Simulated annealing design: Case 3.

Optimizer	Error	NFE's
Downhill Simplex	6.26	960
Simulated Annealing	6.20	2 800
Differential Evolution	3.36	57 000

Table 4.4: Comparison of optimization algorithms: Case 3.

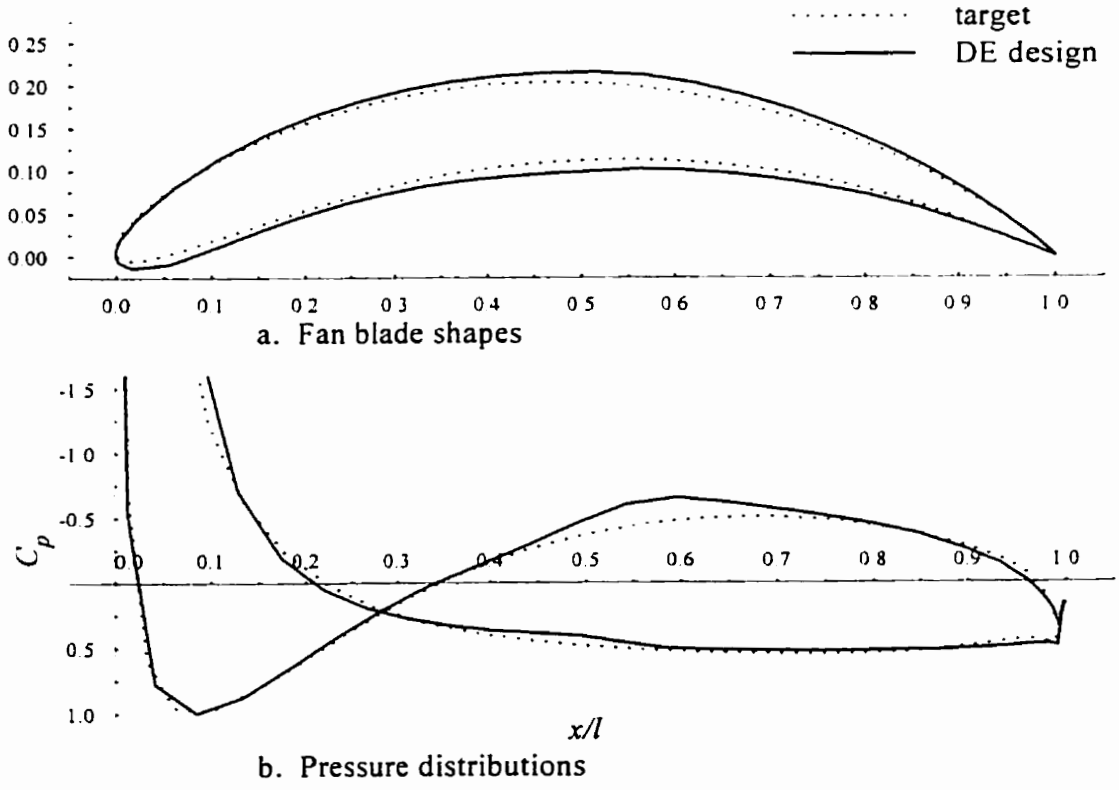


Figure 4.10: Differential evolution design: Case 3.

Optimizer	Error	NFE's
Downhill Simplex	4.42	2 300
Simulated Annealing	2.92	1 600
Differential Evolution	1.16	188 000

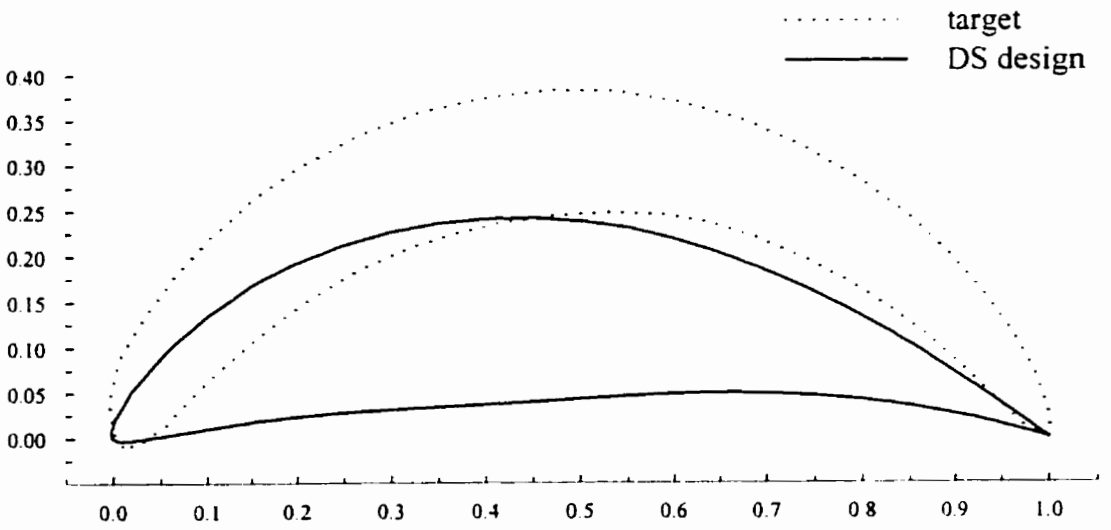
Table 4.5: Comparison of optimization algorithms: Case 4.

away from the nose, and came extremely close to the actual geometry of the target blade. Differential evolution still requires by far the most number of function evaluations, but the other two methods were completely unreliable. The fan blades they designed generate pressure distributions that only faintly resemble the target.

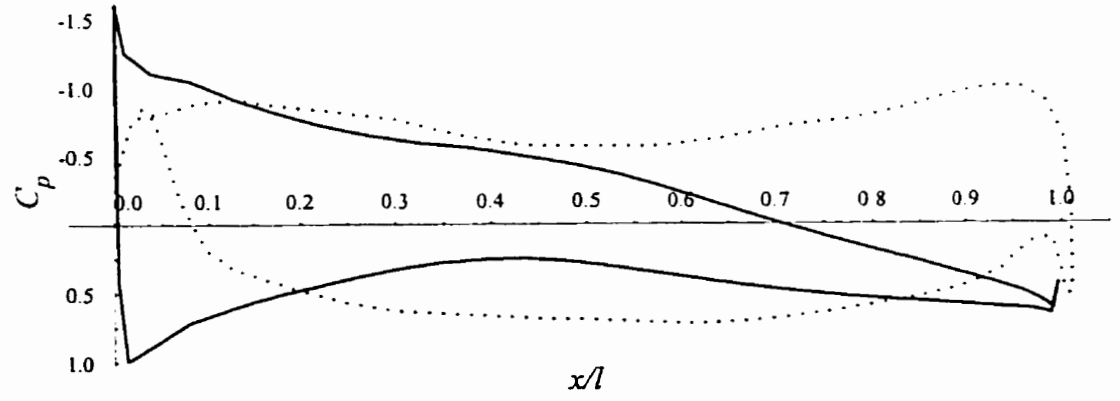
4.1.5 Case 4: Highly Cambered Airfoil

The final test case uses the 112° cambered airfoil, used to test the panel method flow solver in section 3.1.3. The pressure distribution is calculated for $\beta_1 = 50°$, $\lambda = 0$, $t/l = 0.5899644$, $W_1 = 1.0$, and then used as the inverse design target.

After the 10^{-5} tolerance was reached, the algorithms achieved the results summarized in Table 4.5. Downhill simplex again provides a poor representation of the optimal shape (Fig. 4.11), being unable to increase the degree of the camber curve as much as is required. Simulated annealing does better at matching the camber of the target (Fig. 4.12), but can't sufficiently smooth out the final shape. As shown in Fig. 4.13, differential evolution once again performs superbly. Its designed pressure distribution agrees quite well with the target except in the regions near the nose and tail, which have sharp edges which are very difficult to match. Even with the deviation in those regions, the designed shape provides an excellent approximation to the target fan blade.

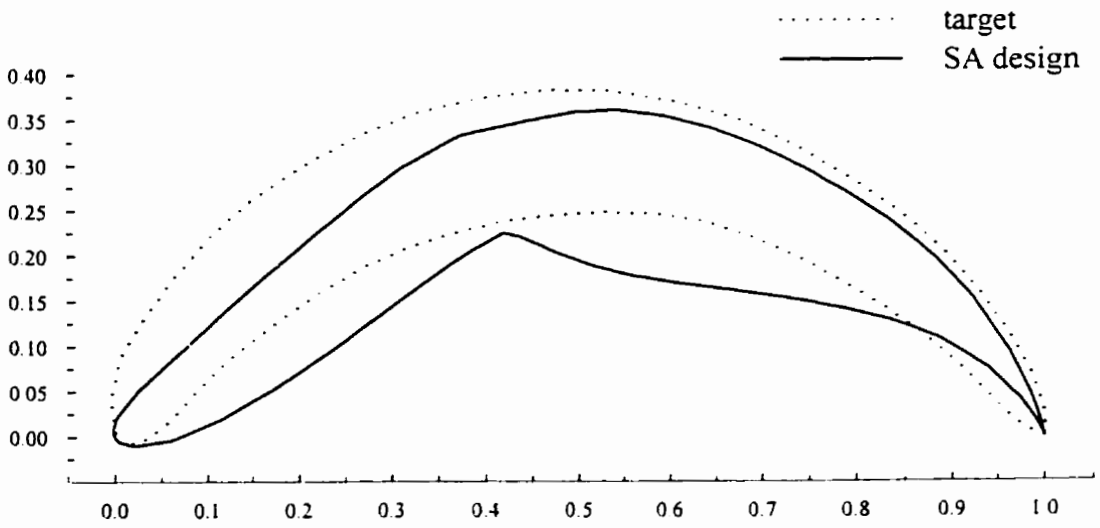


a. Fan blade shapes

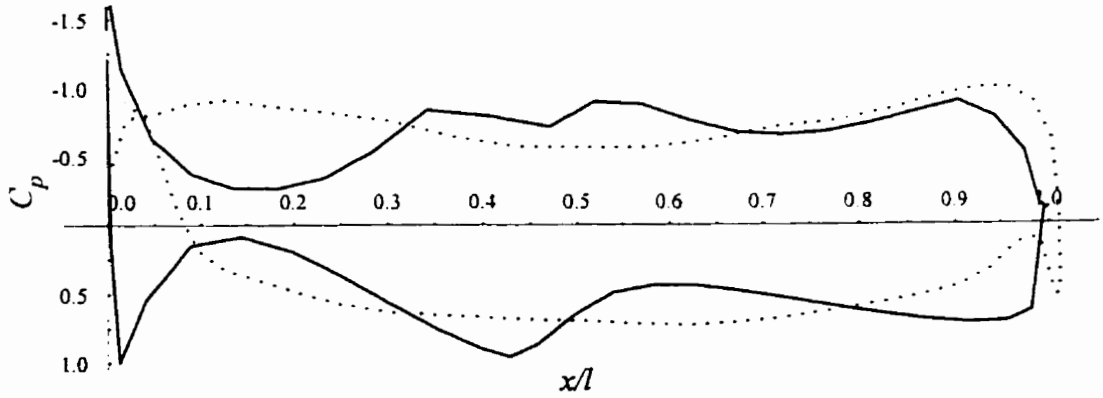


b. Pressure distributions

Figure 4.11: Downhill simplex design: Case 4.

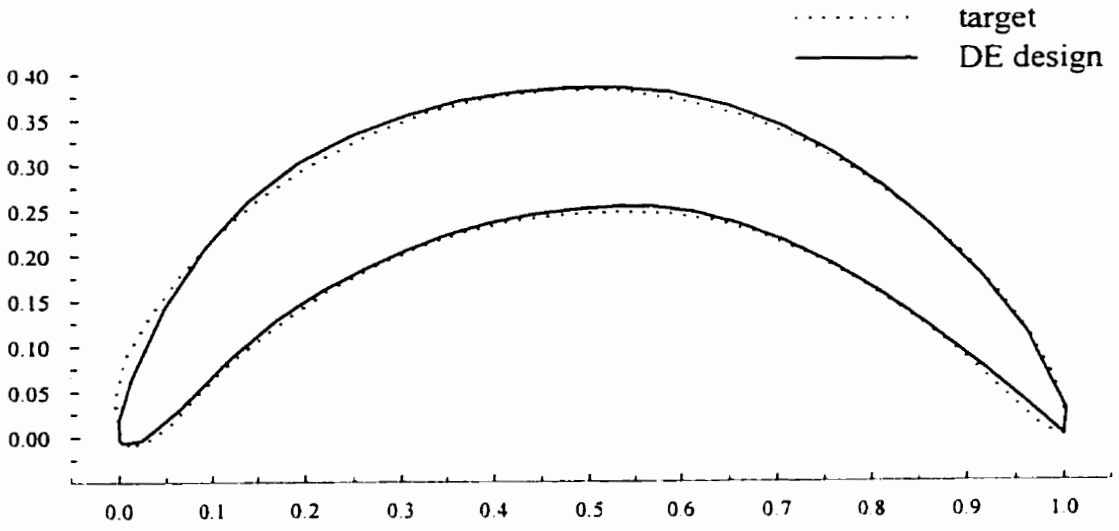


a. Fan blade shapes

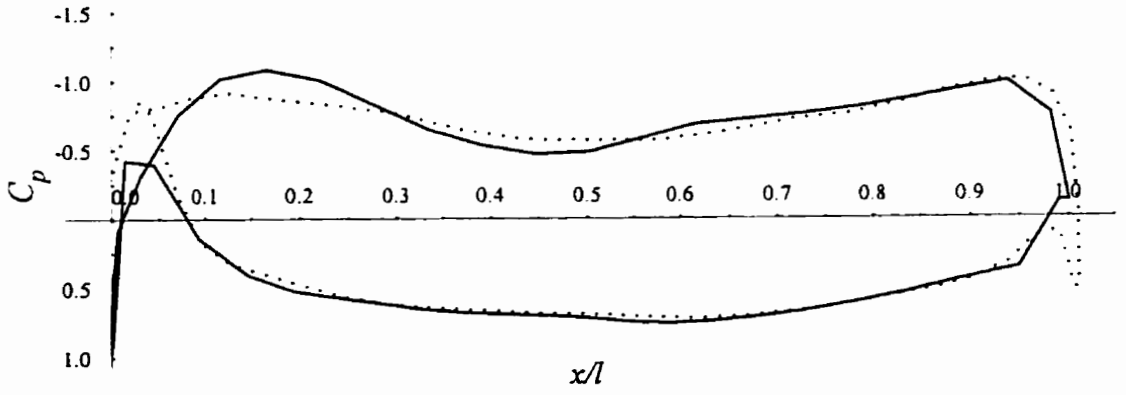


b. Pressure distributions

Figure 4.12: Simulated annealing design: Case 4.



a. Fan blade shapes



b. Pressure distributions

Figure 4.13: Differential evolution design: Case 4.

This test case was taxing even for differential evolution. It is suspected that the use of more data points for both the target distribution and the test shapes would allow this optimizer to perform even better. For the purpose of algorithm comparison, however, the coarse resolution is sufficient, allowing the tests to be run as quickly as possible. Differential evolution was able to match quite closely the target shape. The other two algorithms were once again woefully inadequate in comparison. For the first time, the downhill simplex method was the least effective. It is no coincidence that in this test case the initial geometry was the most different from the target geometry. This hill-climbing algorithm is finally showing its true colours as a local minimum finder.

4.1.6 Comparison of Optimization Algorithms

The number of function evaluations and the final error for each test case and each algorithm are summarized in Table 4.6. In all cases studied, the Downhill Simplex (DS) and Simulated Annealing (SA) methods were very quick in obtaining a converged solution compared to Differential Evolution (DE). In only the first test case, however, was at least one of the quicker methods able to match the effectiveness of the genetic algorithm. In this test case, the initial shape provided to the algorithms was very close to the actual optimum. In all other cases, differential evolution was much more effective in producing a shape which closely modelled the target flow.

The downhill simplex method is unable, in general, to find a global minimum. It must be mentioned, though, that if the initial simplex brackets the optimal shape, it will quickly converge to that solution. It would thus be an effective algorithm if the approximate shape of the optimal solution is known in advance. It is also good at quickly improving

	NFE's			Error		
	DS	SA	DE	DS	SA	DE
Case 1:	1700	2 600	47 000	.095	.569	.096
Case 2:	1 200	1 700	57 000	.396	.559	.195
Case 3:	960	2 800	57 000	6.26	6.20	3.36
Case 4:	2 300	1 600	188 000	4.42	2.92	1.16

Table 4.6: Performance summary of the optimization algorithms.

on the initial geometry. Thus, if a designer merely desires an improvement on an already efficient shape, downhill simplex would be an excellent choice.

Simulated annealing, in addition to being ineffective in general, carries with it the additional hassle of trying to find the best annealing schedule. This can only be done by experiment, and is a rather arduous and frustrating process. Simulated annealing is thus not suggested for use in aerodynamic optimization. If a quick and dirty strategy is necessary, downhill simplex is just as effective, is more efficient, and is much easier to implement.

In each of the case studies examined here, differential evolution was able to find the best shape. It solved even the most difficult inverse design cases, when shapes designed by the other optimization strategies were really no solutions at all. By far, it requires the most CPU time. Even on a slow PC, however, it was able to solve the toughest problems with an overnight run. Because it is so much more effective for difficult design cases, differential evolution is chosen as the "optimal" optimization strategy of the three discussed here.

4.2 Liebeck Fan Blade Design

A Liebeck pressure distribution is used as the target of the inverse design problem. This will demonstrate the technique and verify the effectiveness of the computer code. An interesting difficulty encountered and solved during the design process leads to an important conclusion about optimal pitch/chord spacing of the fan blades in the fan.

This section is intended to illustrate the abilities of the aerodynamic shape optimization package developed in this thesis. It will not catalogue a large number of optimally shaped blades. This may be done by any designer using the code.

4.2.1 Problem Definition

The purpose of an inverse design problem is to design the geometry of a fan blade that will generate a given pressure distribution. However, the shape of the fan blade is only one contributor to the overall performance of a fan. The physical structure of the fan will also play a big role. Modifying the stagger angle and pitch/chord spacing alone will generate a different pressure distribution around the blade. The fan performance parameters that determine the volume and direction of the air flow, however, are the inlet and outlet flow angles. These latter two angles, then, are used as input design parameters. The stagger angle and pitch/chord ratio will be designed along with the shape in the optimization process.

Thus a succinct definition of the inverse design problem is as follows. Given a desired pressure distribution, an inlet angle, and an outlet angle; design the optimal blade shape, stagger angle, and pitch/chord ratio for the fan.

The Liebeck pressure distribution family has been demonstrated to generate excellent lift coefficients. Some properties of the family will be described below (section 4.2.2). One Liebeck pressure distribution is chosen as the target to demonstrate a solution to the inverse design problem. Several fan blade shapes, designed for different inlet and outlet angles of the resultant flow, will be developed to conform to the given distribution.

Initially, designs used a maximum panel length of Lmax = .05. The resultant meshes were sometimes too coarse to be sure of the accuracy of the results. Decreasing Lmax by too much, however, dramatically increased the computational time of the flow solver. A good compromise between precision and speed was obtained by using Lmax = 0.025 for all test runs of the design process. After a solution had been obtained, the flow was recalculated around the designed shape but with a much finer mesh, using Lmax = 0.001. In this way the solution was verified, and a smoother shape was obtained for plotting the results.

4.2.2 Design Parameters

The most important input design parameter is the pressure distribution itself. Liebeck (1973) developed a family of optimal velocity distributions for wing airfoils. This family satisfies three criteria. First, the boundary layer does not separate. Second, the corresponding airfoil shape is practical and realistic. Third, the maximum possible lift is obtained. These optimal distributions utilized the Stratford (1959a) pressure recovery distribution, which avoids separation by a specified margin along its entire length. In principle, this recovers a given pressure rise in the shortest possible distance. Alternatively, it can be interpreted as recovery of the maximum pressure rise in a given distance.

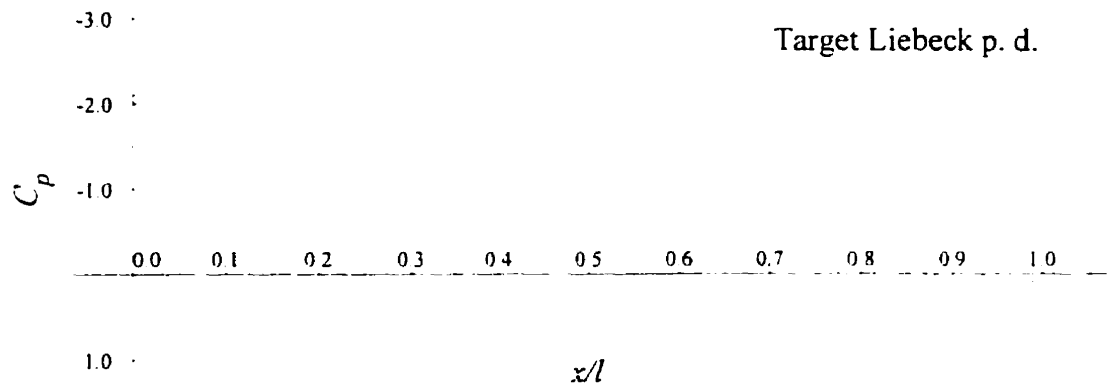


Figure 4.14: Discrete target Liebeck pressure distribution.

Using the calculus of variations, Liebeck was able to solve for the suction, or upper, surface pressure distribution that would provide the maximum lift for any Reynold's number parameter, Re_∞. This consisted of a flat rooftop followed by a Stratford recovery distribution. However, the corresponding optimal velocity distribution would not provide an airfoil shape without some modifications. Among these, the upper surface rooftop was reshaped to allow for operation at a range of angles of attack. Also, a boundary-layer transition ramp was added at the rooftop peak. The resulting velocity distributions are of course no longer purely optimal, but it has been demonstrated, both theoretically and experimentally, that they do provide excellent lift coefficients.

Liebeck also provides some pressure distributions. The one used as the target for all design problems discussed here is designed for $\mathrm{Re}_\infty = 3 \times 10^6$ and angle of attack $\alpha = 9.2°$. Liebeck's distributions are continuous, but a discrete version is necessary for the current inverse design optimization problem. Fig. 4.14 was obtained by plotting directly from Fig. 9 in Liebeck (1973), using a coarse grid on an enlarged figure.

Note that the distribution of points is not uniform. Since the suction surface distribution is more critical in the blade's performance (Walker, 1976), it contains 40 points compared to 20 on the blade's lower surface. Also, the pressure recovery region is the most important part of the suction surface. Thus, of the 40 data points on this upper surface, 30 are dedicated to following the Stratford distribution. By distributing the data points in this manner, the error calculated by the objective function is essentially weighted more heavily in these most consequential regions.

It should be noted that the pressure coefficients in Liebeck's paper are scaled with respect to W_∞. The Lewis flow solver used, however, scales the pressure coefficient with respect to W_1. If the velocity $w(x)$ is known, the pressure coefficient computed with respect to constant velocity W_α is

$$C_{p_\alpha}(x) = 1 - \left(\frac{w(x)}{W_\alpha}\right)^2. \tag{4.1}$$

The conversion required is from C_{p_∞}, the Liebeck pressure coefficients, to C_{p_1}, the pressure coefficients used by the flow solver. The velocity at infinity, W_∞, is the vector mean of the inlet and outlet velocities

$$W_\infty = \frac{W_1 \cos\beta_1}{\cos\beta_\infty}, \tag{4.2}$$

where β_∞ is given by equation (3.19):

$$\tan\beta_\infty = \frac{1}{2}\left(\tan\beta_1 + \tan\beta_2\right).$$

Using equations (4.1) and (4.2)

$$C_{p_1} = 1 - (1 - C_{p_\infty})\left(\frac{\cos\beta_1}{\cos\beta_\infty}\right)^2. \tag{4.3}$$

The correctly scaled target distribution thus depends on the inlet and outlet angles, and must be recomputed for each distinct (β_1, β_2) pair.

The problem initially posed was to design eight compressor fan blade shapes corresponding to two inlet-outlet angle pairs, $(\beta_1, \beta_2) = (30°, 0°)$ and $(60°, 0°)$, and four logarithmically scaled pitch chord ratios, $t/l = 1/\sqrt{10}, 1, \sqrt{10}$, and 10, with the inlet velocity fixed at $W_1 = 1$. In order to fix β_1 and β_2, the stagger angle λ was added to the list of control variables, which for testing had only included Bezier curve control points. The first design attempted was for a fan with $\beta_1 = 30°$, $\beta_2 = 0°$, and $t/l = 1$.

4.2.3 Additional Geometric Constraints

Shapes such as those in Figures A.1, A.2, and A.3 were encountered for this first design problem ($\beta_1 = 30°$, $\beta_2 = 0°$, $t/l = 1$), necessitating additional geometric constraints. These shape irregularities did not occur in the testing stage of the algorithm. They are problem dependent, and in this case are caused by the optimal nature of the pressure recovery region of the Liebeck distribution. The development of further constraints may form part of the design process for any other given pressure distribution, so the procedure by which the constraints were determined is described below.

The flow solver makes some simplifying assumptions. For this discussion, the most significant assumption is non-separation of the flow. The panel method will, in general, incorrectly model the flow over irregularities such as the bumps and sharp edges on the airfoils in Figures A.1, A.2, and A.3. Examining the designed Bezier curves separately enables additional constraints to be determined for the difficult regions.

Output files generated by the code include a list of the Bezier control points for

each of the four curves used to define the airfoil shape. Using the parametric equations (3.25), these control points can be plotted together with the thickness and camber curves. A visual inspection can normally determine which control point(s) caused the irregularity in the resultant fan blade shape. From this analysis, an additional constraint can be imposed on the control points which will remove from contention any fan blades exhibiting the observed irregularity.

The designer should not be disturbed if the imposition of one constraint is followed by a shape irregularity in another region. This was the experience in the current design problem, as the Liebeck distribution resisted the design of smooth shapes. One constraint was followed by another in a spiralling convergence toward a realistic solution.

Two irregularities can be observed in the airfoil shown in Fig. A.1a. The leading edge is pointed because the thickness curve is not normal to the x-axis at $x = 0$ (Fig. A.1b). This is caused by the small y-value of the second leading edge thickness control point, t_2 in the figure. In fact it is practically coincident with the first control point, $t_1 = (0, 0)$. Thus the thickness control points were constrained such that the second y-value would be greater than a predifined constant Sep = 0.05. Recall that a constraint violation triggers the use of the penalty function without calculating the flow (section 3.4.3).

Also notice the "pothole" on the lower surface of the shape in Fig. A.1a. The cause can be observed in the leading edge camber curve (Fig. A.1c). The second control point, c_2 in the figure, has an x-value very close to that of c_3, but a y-value significantly higher. This creates the sharp dip in the camber curve toward the joint with the trailing edge curve. In addition to creating the pothole, this phenomenon makes it more difficult to

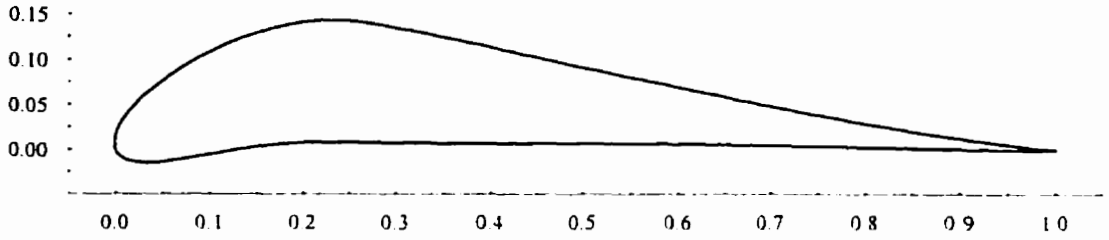
determine the point and value of maximum camber, which is supposed to be at c_4, the joint between the leading and trailing edge camber curves. To resolve this problem, y-values of control points on either side of the camber curve joint were restricted to be smaller than 0.01 above the y-values of the center points. That is, $c_2 - c_4 < 0.01$ and $c_6 - c_4 < 0.01$.

After these constraints were imposed, the next airfoil designed had a bump at the tail (Fig. A.2a). This was caused by the height of the second last control point on the trailing edge thickness curve, t_6 in Fig. A.2b. This point was consequently constrained to be lower than the center control point (t_4 in Fig. A.2b).

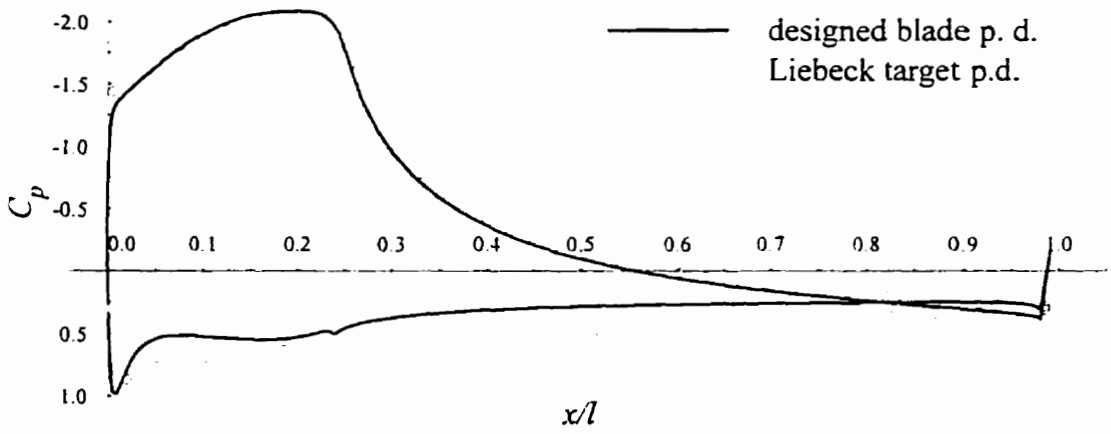
In Fig. A.3a, the bump on the lower edge was caused by the thickness curve (Fig. A.3b). In the trailing edge thickness curve, the second control point, t_5, is too close to the center point, t_4, to prevent t_6 from quickly dragging the curve down. This creates a sharp second order discontinuity in the thickness curve at the joint between the Bezier curves, t_4. This was smoothed out by doubling the minimum distance between t_4 and t_5: $t_5 - t_4 < 2(\text{Sep})$.

4.2.4 Liebeck Design 1

The additional constraints discussed above enabled the design of a smooth Liebeck fan blade shape. The airfoil and its pressure distribution are shown in Fig. 4.15. The L_2-error norm is 0.861, but notice the excellent agreement with the Stratford pressure recovery region, which is the most important feature of the flow. The deviation of the pressure distributions at the leading edge and in the lower surface are relatively insignificant. In fact, this fan blade shape is quite similar to the wing airfoil designed by Liebeck for the same pressure distribution. The stagger angle necessary for the given flow is $\lambda = 3.338°$.



a. Designed fan blade shape



b. Pressure distribution comparison

Figure 4.15: Liebeck Design 1 ($\beta_1 = 30.\ \beta_2 = 0,\ t/l = 1.0$).

The Bezier curve control points are given in Table 4.7, and the thickness and camber curves themselves are shown in Fig. A.4. Maximum thickness is 13.53%, occurring at 22.98% of the chord. Maximum camber is 7.55%, at 23.73% chord.

Notice the small, sharp bump on the lower edge of the pressure distribution in Fig. 4.15b, at roughly $x = 0.23$. One consequence of joining two Bezier curves is the fact that the joint is only first order continuous. The discontinuities in the higher order derivatives will thus negatively impact the smoothness of the curve at that point. In this first design, the thickness curves join at $x = 0.230$, and the camber curves join at $x = 0.237$ (Table 4.7).

	Thickness		Camber	
	x	y	x	y
Leading control points	0	0	0	0
	0	0.05	0.030261	0.0085283
	0.178835	0.0676387	0.116721	0.0755104
	0.229758	0.0676387	0.237257	0.0755104
Trailing control points	0.229758	0.0676387	0.237257	0.0755104
	0.329759	0.0676387	0.287258	0.0755104
	0.601415	0.0210174	0.877770	2.67×10^{-7}
	1	0	1	0

Table 4.7: Bezier control points - Liebeck Design 1

These points are so close that the discontinuities are compounded. While not visible on the blade (Fig. 4.15a) they do show up in the flow solution.

4.2.5 Optimal Pitch/Chord Ratio

Having produced a Liebeck fan blade designed for $\beta_1 = 30°$, $\beta_2 = 0°$, and $t/l = 1$, attempts were made to design shapes for the remaining three pitch/chord ratios, $t/l = 1/\sqrt{10}, \sqrt{10}$, and 10, with the 30° inlet angle. In all cases, however, the optimization code could not find a solution. In each of the three failures, the optimizer converged to errors greater than 3.5 (see Table 4.8, in which the one solution is also included for comparison). To see any similarity between a 60-point target pressure distribution and that of the designed blade, the L_2-error norm must be close to, and preferably less than, one. The final pressure distributions in these cases exhibited none of the characteristics of the Liebeck distribution.

In an attempt to explore the reason for these failures, the flow around the fan blade designed for $t/l = 1$ was computed for the remaining three pitch/chord ratios. Results for $t/l = \sqrt{10}$ and 10 are shown in Fig. 4.16. Notice that the magnitude of the pressure on

β_1	β_2	t/l	L_2-error norm
30	0	$1/\sqrt{10}$	3.568
30	0	1	0.861
30	0	$\sqrt{10}$	7.281
30	0	10	23.119

Table 4.8: Errors for design problems with fixed pitch/chord.

the suction surface increases as the pitch/chord ratio increases. This will actually improve the lift coefficient of the fan blade. Also, the resultant pressure distributions are scaled versions of the target Liebeck distribution. In fact, they closely resemble other members of the family of pressure distributions designed by Liebeck. They will be nearly optimal for other flow conditions. Thus, not only is there no solution for selected blade spacings, but the designed blade also appears to solve other distinct inverse design problems targetting an alternate set of pressure distribution and pitch/chord parameters.

From these results, it is postulated that solutions will not exist for all sets of pressure distribution, β_1, β_2, and t/l. It is further postulated that an optimal pitch/chord ratio will exist for an inverse design problem targetting a given pressure distribution, inlet angle, and outlet angle. To verify this claim, t/l is optimized alongside λ and the geometric variables. The success of the modified design optimization scheme is demonstrated below.

It is in fact well known in the aerodynamic engineering community that optimal pitch/chord ratios exist to minimize the losses of the blading (Stewart & Glassman, 1974). Until now, the claim has been supported experimentally and with very simple modelling. The current design scheme not only verifies the experimental results, but actually provides a mechanism for finding this optimal blade spacing.

It was a simple matter to modify the code to add t/l to the control variable list.

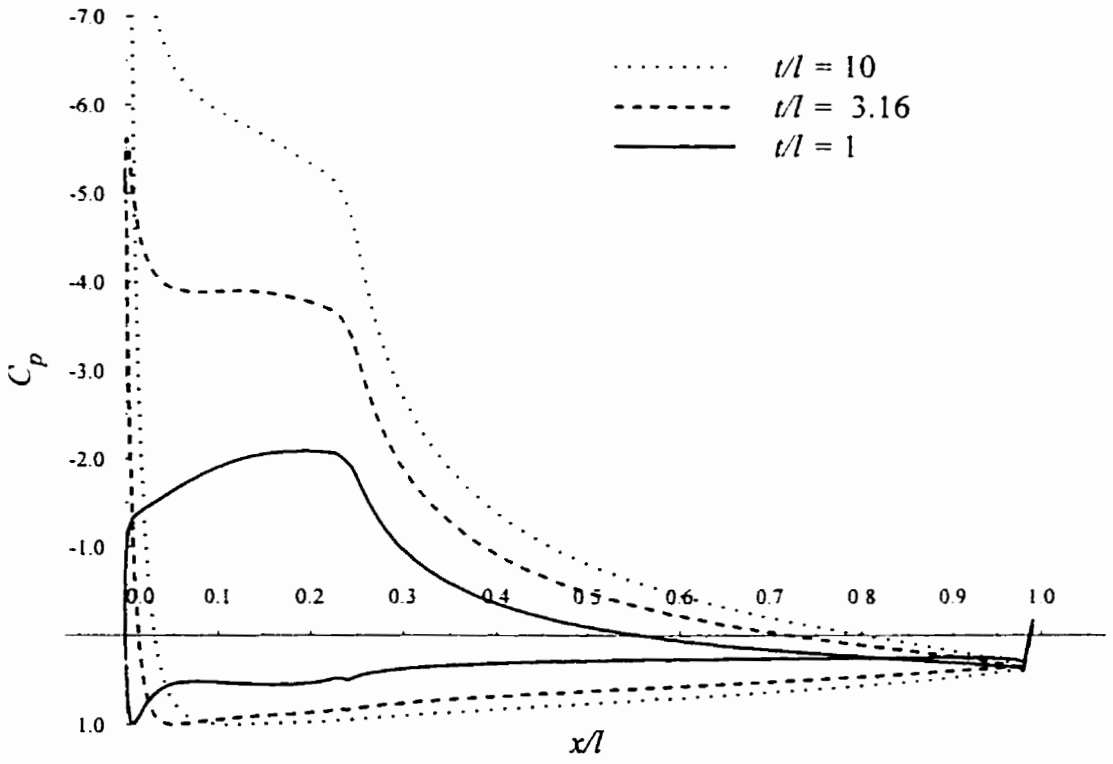


Figure 4.16: Performance of Liebeck Design 1 for different pitch/chord ratios.

	Thickness		Camber	
	x	y	x	y
Leading control points	0	0	0	0
	0	0.05	0.034470	0.0072003
	0.161147	0.0531654	0.126967	0.0865875
	0.211147	0.0531654	0.241969	0.0865875
Trailing control points	0.211147	0.0531654	0.241969	0.0865875
	0.311148	0.0531654	0.291970	0.0865875
	0.327274	0.0380168	0.936801	0.0102217
	1	0	1	0

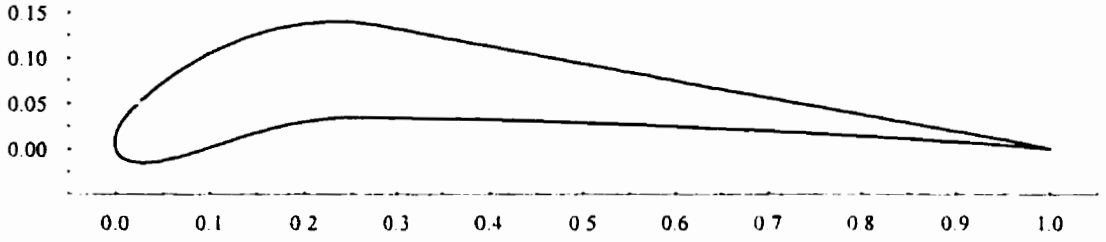
Table 4.9: Bezier control points - Liebeck Design 2

The only constraint is $t/l > 0$, and this was added to the list of restrictions dealt with by the penalty function. The two design cases, $\beta_1 = 30°, 60°$, were then repeated, with t/l now variable.

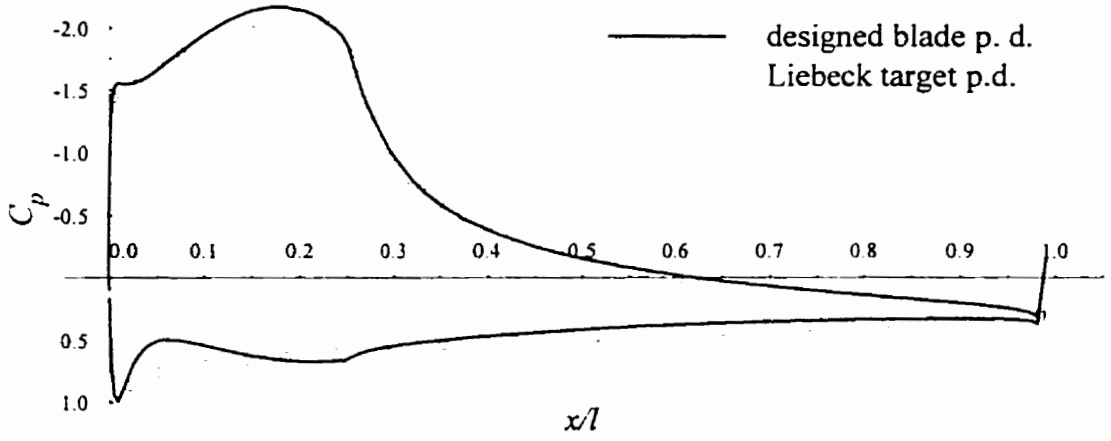
4.2.6 Liebeck Design 2

In the first design case, $\beta_1 = 30°$, the optimal pitch/chord ratio was found to be $t/l = 1.183$. Optimizing t/l allowed the error to be driven down to 0.619, compared to 0.861 when t/l was fixed at 1.0. The resultant cross-sectional fan blade shape and corresponding pressure distribution are shown in Fig. 4.17. The stagger angle necessary for the given flow is $\lambda = 4.734°$. The Bezier curve control points are given in Table 4.9. Thickness and camber curves are shown in Fig. A.5. Maximum thickness is 10.63%, occurring at 21.11% of the chord. Maximum camber is 8.66%, at 24.20% chord.

The similarities between the first two designed blades can be seen in Fig. 4.18. Also note the similarities in the Bezier curve control points, Tables 4.7 and 4.9. The closeness of the optimal spacing $t/l = 1.183$ for Liebeck Design 2 to $t/l = 1.0$ was what



a. Designed fan blade shape



b. Pressure distribution comparison

Figure 4.17: Liebeck Design 2 ($\beta_1 = 30$, $\beta_2 = 0$, $t/l = 1.183$).

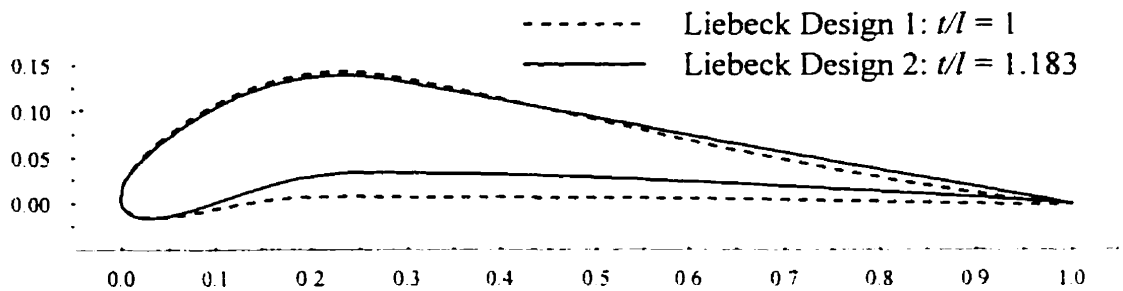


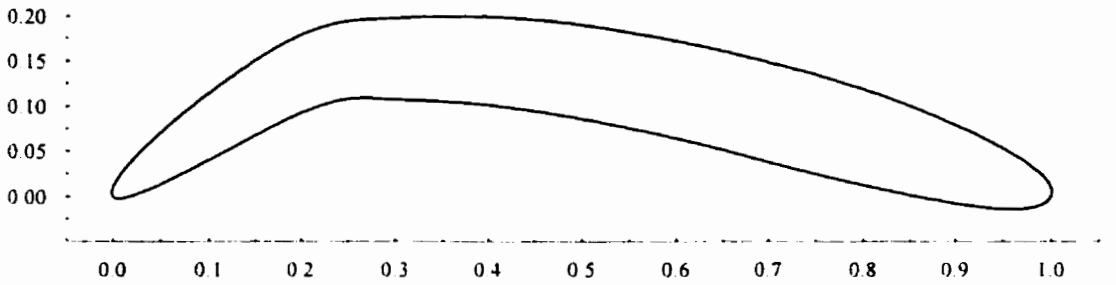
Figure 4.18: Comparison of first two Liebeck fan blades.

enabled the design optimization code to find the first solution, Liebeck Design 1 (section 4.2.4).

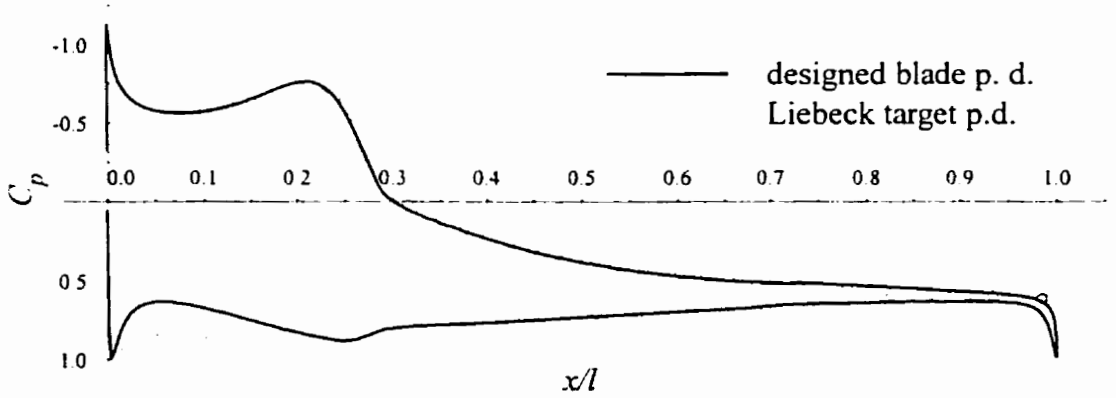
4.2.7 Liebeck Design 3

Liebeck Design 3a was the first blade designed for the second case, $\beta_1 = 60°$. The optimal pitch/chord ratio was found to be $t/l = 0.560$. The cross-sectional fan blade shape and corresponding pressure distribution are shown in Fig. 4.19. The L_2-error norm between the Liebeck target and the designed distributions is 0.458. The stagger angle necessary for the given flow is $\lambda = 17.701°$. The Bezier curve control points are given in Table 4.10. Thickness and camber curves are shown in Fig. A.6. Maximum thickness is 10.72%, occurring at 71.25% of the chord. Maximum camber is 15.23%, at 29.27% chord.

Unfortunately, the potential flow solver used does not account for separation. There will definitely be separation because of the rounded trailing edge. This may only occur in the last 5% of the chord, but would still cause performance losses. Also, the Stratford distribution is actually designed for flat plates. It considers only the pressure gradient, and does not account for surface curvature. There is thus a two-fold modelling



a. Designed fan blade shape



b. Pressure distribution comparison

Figure 4.19: Liebeck Design 3a ($\beta_1 = 60$, $\beta_2 = 0$, $t/l = 0.560$).

	Thickness		Camber	
	x	y	x	y
Leading control points	0	0	0	0
	0	0.050093	0.207059	0.162334
	0.582445	0.053605	0.242674	0.152335
	0.712508	0.053605	0.292675	0.152335
Trailing control points	0.712508	0.053605	0.292675	0.152335
	0.813005	0.053605	0.394230	0.152335
	0.999999	0.041572	0.527354	0.162334
	1	0	1	0

Table 4.10: Bezier control points - Liebeck Design 3a.

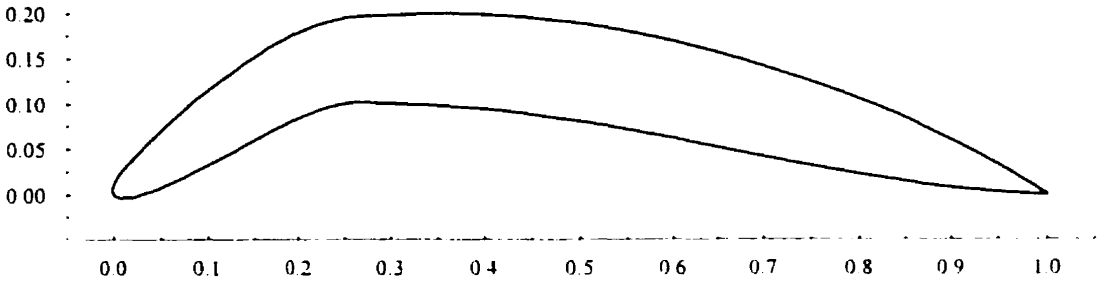
problem plaguing this design case.

Attempts will still be made to design a more reasonably shaped blade, but the results will not be entirely satisfactory. The rounded edge is caused by the second last thickness control point. It's x-value is so close to 1 that the thickness curve is normal to the x-axis at the trailing edge. In an attempt to design a blade with a sharp trailing edge, a second run was performed constraining the second last thickness control point to be left of 0.95, resulting in Liebeck Design 3b (Fig. A.7). The resulting blade is no longer rounded at the trailing edge, but is still not well-streamlined.

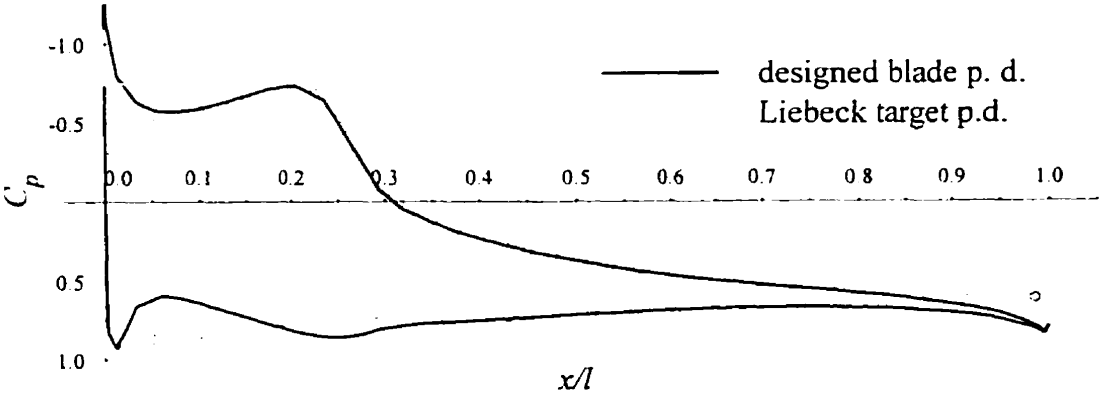
Also, notice that the maximum thickness of Design 3a is at 71.25% of the chord. This point is typically desired to be in the front half of the blade. Thus a constraint was added requiring the juncture of the leading and trailing thickness curves to occur left of $x = 0.5$. For this third run, the constraint used above to force a sharp trailing edge was removed. This resulted in Liebeck Design 3c (Fig. A.8), a shape not significantly different from the original Design 3a.

A combination of these constraints was then attempted, requiring maximum thickness to occur at less than 50% of the chord, and requiring the second last control point to be left of $x = 0.9$ this time. The resulting Liebeck Design 3d (Fig. A.9) now has a cusped trailing edge. This was caused by the horizontal closeness the middle two control points on the trailing edge thickness curve. These points were then constrained to have x-values separated by at least 0.05.

This final result would seem to be the best $\beta_1 = 60°$ design yet. For this Liebeck Design 3e, the optimal pitch/chord ratio was found to be $t/l = 0.553$. The cross-sectional



a. Designed fan blade shape



b. Pressure distribution comparison

Figure 4.20: Liebeck Design 3e ($\beta_1 = 60$, $\beta_2 = 0$, $t/l = 0.553$).

fan blade shape and corresponding pressure distribution are shown in Fig. 4.20 . The L_2-error norm between the Liebeck target and the designed distributions is 0.528. The stagger angle necessary for the given flow is $\lambda = 16.951°$. The Bezier curve control points are given in Table 4.11. Thickness and camber curves are shown in Fig. A.10. Maximum thickness is 10.70%, occurring at 49.99% of the chord. Maximum camber is 14.88%, at 26.93% chord.

These last blades are designed to be very highly loaded as a consequence of the large disparity between the inlet and outlet angles. Notice that the drooped nose is a feature

	Thickness		Camber	
	x	y	x	y
Leading control points	0	0	0	0
	0	0.050000	0.068562	0.041329
	0.449998	0.053482	0.185563	0.148806
	0.499999	0.053482	0.269301	0.148806
Trailing control points	0.499999	0.053482	0.269301	0.148806
	0.673499	0.053482	0.399661	0.148806
	0.860706	0.045023	0.517970	0.158795
	1	0	1	0

Table 4.11: Bezier control points - Liebeck Design 3e.

shared by all designs. Such a leading edge flap is not uncommon in such circumstances. Also notice the sharp pressure rise in the trailing region of each of Designs 3a - 3e. This may indicate the region of separation. It is possible that more acceptably shaped blades could be designed if the error was weighted more heavily in this trailing region.

Chapter 5

Conclusions

In this work, an aerodynamic shape optimization software package has been developed which will design cross-sectional fan blade shapes using an inverse technique. Several noteworthy comments can be made. Any inverse design algorithm must contain a flow solver, a shape definition module, and an optimizer. The algorithm developed here has provided a mechanism for evaluation of the specific component parts chosen. The suitability for aerodynamic optimization of the following components will be discussed: the surface vorticity potential flow solver, Bezier curve geometry parametrization, and the three optimization algorithms used. During the design process, an unexpected but very interesting feature of fans was discovered, namely that an optimal design will likely also include a unique pitch/chord spacing of the blades. It is suggested that the inverse design algorithm developed here may be a valuable tool in the verification of analytical attempts to determine this optimal spacing. Finally, properties of the actual fan blade shapes designed will also be considered.

The surface vorticity panel method is undoubtedly extremely quick, making it a prime candidate for use in aerodynamic optimization. Flow calculations for roughly 50 000 shapes were able to be performed in only five hours on an old Intel machine. The runtime could undoubtedly have been reduced to well under an hour on a workstation. Sophisticated CFD flow solutions, on the other hand, have been known to take days for one shape. This is clearly unacceptable in an optimization strategy in which many designs must be compared aerodynamically. Even with the simplifications required for the panel method to achieve its efficiency, the technique was sufficient to enable the design of complex, but very realistic shapes. The first two Liebeck fan blades designed, for example, are very similar in shape to the wing airfoils designed by Liebeck himself, airfoils whose characteristics have been fully verified in wind tunnel tests.

However, when shapes exhibit extreme geometries, such as regions of obvious separation and discontinuous surfaces, the panel method will break down and become unreliable. In some design cases, this shortcoming was overcome by introducing geometric constraints to force the solutions into a reasonable domain. In other cases, though, difficult regions of the resulting shapes could not be removed, and the results might possibly be compromised. Thus any designs should be verified, at the very least, by a more sophisticated CFD model. Despite their limitations, panel methods are extremely effective as filters to allow candidate solutions in optimization strategies. In this respect, the surface vorticity method provides an admirable low cost technique for exploring the solution space.

Bezier curves provide the aerodynamic designer with an excellent tool for shape definition. The surface can be controlled by only a few control points, and the surface gen-

erated is guaranteed to be smooth and without wild oscillations. Unacceptable geometries can be easily removed from contention by appropriately constraining the control points. This enables not only the elimination of some extreme geometries, as discussed above, but also the incorporation of specific structural or manufacturing constraints into the design process. Furthermore, the geometry identified can easily be exported to sophisticated flow analysis programs that accept spline-based surface definition.

Differential Evolution is a fairly new genetic algorithm which has very promising applications in aerodynamic design. In all inverse design cases studied in this work, differential evolution was the most effective optimizer. It consistently produced a shape which closely modelled the target flow, and was able to solve some challenging problems even when other optimizers were incapable of doing so. By far, it is the most costly of the three techniques compared, typically requiring between 50 000 and 100 000 test flow calculations to design the final shape. Even on a slow PC, however, it was able to solve the toughest problems with an overnight run.

For aerodynamic shape optimization problems, genetic algorithms are proving to be the best strategies. Similar results were reported by Obayashi and Tskukahara (1995). Although the specific aerodynamic optimization problem and the optimizers compared were different from those examined here, a genetic algorithm was also able to outperform all other approaches. They do require many more function calls than more conventional algorithms. As a result, however, they are able to seek out the global minimum, even in formidable landscapes containing many local minima. With computational power ever-increasing, soon even the slow rate of convergence will not hold them back.

One significant contribution of the current inverse design algorithm is the ability to determine optimal pitch/chord spacing of blades in a fan. Attempts were made as early as 1945 (Zweifel, 1945) to determine analytically the solidity (inverse of the pitch/chord ratio) that would minimize losses in a cascade. As noted in Stewart and Glassman (1974), however, the experimental results used to verify the analysis "are usually obtained by using a given blade shape and varying the spacing. Thus, the blade shape and resultant velocity distribution cannot be optimized for each solidity, and the significance of such correlation is somewhat clouded." The strength of the current work lies precisely in addressing this issue. With computer modelling, not only can the cost otherwise generated by experimentation be reduced, but more general results can actually be obtained. It should be possible, then, to use this or similar code to verify the analytical predictions of optimal solidity for given flow conditions.

Two inverse design problems were attacked using the aerodynamic shape optimization package. For the first case, a fan blade shape was designed exhibiting a Liebeck pressure distribution for a flow with a 30° inlet angle and a 0° outlet angle. The required stagger angle and pitch/chord ratio were also determined. This shape, Liebeck Design 2, can be seen to exhibit many of the characteristics of wing airfoil shapes designed by Liebeck himself. Those Liebeck shapes have been built and tested in wind tunnels. It is fairly certain, then, that this designed blade will perform as required.

The same pressure distribution was used to design a shape for a 60° inlet angle and a 0° outlet angle. This is thus a much more highly loaded blade, and a more difficult design case as a result. The resultant family of shapes, Liebeck Designs 3a - 3e, are not

as streamlined as Design 2. It is suspected that separation will occur near the ends of at least those designs with a rounded trailing edge. The losses that might occur because of this separation are unknown. All designs share the characteristic nose flap which is sometimes seen on highly loaded fan blades. Also, the optimal blade spacing is fairly close. These features will attempt to deflect the flow to the extent required. However, the actual performance of any of these last designs cannot even be proposed without a more sophisticated flow analysis.

Chapter 6

Recommendations

Future work in this field could follow a number of different directions. Attempts could be made to improve the efficiency of the algorithm. These might use different optimizers, or a combination of optimization techniques. The relationship between the optimal pitch/chord ratio and other design parameters could be explored further, and attempts could be made to evaluate existing analytical predictions of this optimal value. Further analysis of the blades designed should be performed to get an accurate idea of their actual performance. The effects of and reasons for the latter designs' predisposition toward rounded trailing edges would be of special interest. Also, additional blades can be designed and compared using alternative pressure distributions.

Alternate optimization routines might be as effective as but more efficient than differential evolution. Gradient-based techniques have proven to be effective in other problems, but were not used here because of the penalty function. The penalty function could be replaced by geometry modification, and then a gradient-based technique tested, using

finite difference methods to compute the derivatives. It is anticipated that differential evolution with the penalty function will still be more effective. Gradient techniques are still essentially local search methods, although they are generally more efficient than the one used here. It is likely that they would also become trapped in a local minimum somewhere in the complicated landscape typical of aerodynamic objective functions. A better place to start looking would probably be among other genetic algorithms.

It might be possible take another approach to improving the optimization efficiency while maintaining the effectiveness generated by the differential evolution technique. Some combination of optimization techniques might be attempted. For example, differential evolution could be used until the associated error doesn't change by more than some small value, after a given number of generations. The resulting shape could then be used as the initial geometry in the quicker downhill simplex routine. Or the best $N + 1$ shapes from the differential evolution solution could be chosen as the initial simplex. With some experimentation, it might be found that differential evolution would sufficiently bracket the solution after a given point, enabling the downhill simplex method to find the global minimum if it is able to start from that point.

The inverse design algorithm has demonstrated an ability to find optimal pitch/chord spacings. The relationship between this ratio and other parameters, such as the lift coefficient and Reynold's number, could be explored further. Also, there have been some attempts at predicting the optimal spacing analytically. As noted in Stewart and Glassman (1974), however, the experimental results used to verify the analysis are not entirely satisfactory because they use fixed blade shapes and are unable to optimize the velocity

distribution. It should be possible to use this code, or a modification of it, to verify the analytical predictions of the optimal pitch/chord ratio for given flow conditions. This field of study is still open, and might benefit from the design optimization model developed here.

A full CFD analysis should be performed on the Liebeck fan blades designed in this thesis. This is expected to verify the results of Liebeck Designs 1 and 2. For Designs 3a - 3e, it would be particularly interesting to know if and where separation occurs, and the corresponding effect on the pressure distribution. If the more sophisticated analysis verifies the results, it could be followed by experimental testing.

In the second design case, Liebeck Designs 3a - 3e showed a propensity toward a rounded edge. It would be fascinating to explore the reasons for this curious design. The causes may include the fact that the prescribed Liebeck pressure distribution doesn't account for curvature of the blade. Or it may simply be that the panel method flow solver is incorrectly modelling the flow. Noting the deviation of the designed pressure distribution in the trailing edge region, a possible approach to correcting the design would be to weight the prescribed distribution more heavily in that region. This could be done by modifying the error function, or by adding data points to the Liebeck distribution in the latter 10% of the chord.

The only blades designed here used a prescribed Liebeck pressure distribution. There are other optimal pressure distribution families that might be considered. For example, Walker (1976) has developed suction surface distributions specifically for fan blades. Walker fan blade shapes could be designed using the same inlet and outlet flow angles as for the Liebeck fan blades designed here. These shapes could be compared and contrasted.

While the Liebeck distributions have been verified and used more extensively, it is possible that improvements on them could be made for fan use. The Walker distributions might provide this enhancement.

Bibliography

[1] Bartels, R. H., Beatty, J. C., and Barsks, B. A., An Introduction to Splines for use in Computer Graphics and Geometric Modelling, Morgan Kaufmann Publishers, Inc. Los Altos, 1987.

[2] Batchelor, G. K., An Introduction to Fluid Dynamics, Cambridge University Press. Cambridge, 1970.

[3] Bender, J. R. Aerodynamic Shape Optimization of Axial Flow Fans, M. Eng. Thesis. University of Manitoba, 1996.

[4] Bezier, P., Numerical Control - Mathematics and Applications, translated by Forrest, A. R. and Pankhurst, A. F., John Wiley & Sons, London, 1972.

[5] Bezier, P., Mathematical and Practical Possibilities of UNISURF, in *Computer-Aided Geometric Design*, Barnhill R. E. and Riesenfeld, R. F. (eds.), Academic Press, New York, 127-152, 1974.

[6] Birckelbaw, L., Inverse Airfoil Design using the Navier-Stokes Equations, AIAA Paper 89-2202-CP, 1989.

[7] Bounds, D. G., New Optimization Methods from Physics and Biology, *Nature* Vol. 329, 215-218, 1987.

[8] Burgreen, G. W., Baysal, O., and Eleshaky, M. E., Improving the Efficiency of Aerodynamic Shape Optimization Procedures, 4^{th} *AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, AIAA Paper 92-4697, Cleveland, 1992.

[9] Coiro, D.P., and Nicolosi, F., Design and Optimization of Glider Components, Technical Soaring, Vol XIX, No. 2, 1995.

[10] Davis, P. J., Interpolation and Approximation, Blaisdell Publishing Company, New York, 1963.

[11] Goldberg, D. E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading, 1989.

[12] Gostelow, J. P., Cascade Aerodynamics, Pergamon Press, Oxford, 1984.

[13] Hartmann D., Optimierung Balkenartiger Zylindeerschalen aus Stahlbeton mit Elastischem und Plastischem Werkstoffverhalten, Ph.D. Thesis, University of Dortmund, 1974.

[14] Hess, J. L. Calculation of Potential Flow about Bodies of Revolution having Axes Perpendicular to the Free-Stream Direction, *Journal of Aerospace Science*, 726-742, 1962.

[15] Hess, J. L. and Smith, A. M. O., Calculation of Potential Flow about Arbitrary Bodies,

in *Progress in Aeronautical Sciences. Vol. 8*, Kuchemann, D. (ed.), Pergamon Press, Oxford, 1-138, 1967.

[16] Hicks, R.M., and Henne, P.A., Wing Design by Numerical Optimization, AIAA Paper 77-1247, 1977.

[17] Holland, J. H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[18] Jacob, K., and Riegels, F. W. The Calculation of the Pressure Distributions over Aerofoil Sections of Finite Thickness With and Without Flaps and Slats, *Z. Flugwiss*, Vol. 11, No. 9, 357-67, 1963.

[19] Kellog, O. D., Foundations of Potential Theory, Frederick Ungar Publishing Company, New York, 1929.

[20] Kirkpatrick, S., Gerlatt, C. D. Jr., and Vecchi, M.P., Optimization by Simulated Annealing, *Science* Vol. 220, 671-680, 1983.

[21] Lewis, R. I., Vortex Element Methods for Fluid Dynamic Analysis of Engineering Systems, Cambridge University Press, Cambridge, 1991.

[22] Liebeck, R. H. and Ormsbee, A. I., Optimization of Airfoils for Maximum Lift, *Journal of Aircraft*, Vol. 7, No. 5, 409-415, September, 1970.

[23] Liebeck, R. H., A Class of Airfoils Designed for High Lift in Incompressible Flow, *Journal of Aircraft*, Vol. 10, No 10, 610-617, October, 1973.

[24] Liebeck, R. H., Design of Subsonic Airfoils for High Lift. *Journal of Aircraft*. Vol. 15, No. 9, 547-561, September, 1978.

[25] Liebeck, R. H., Computational Aerodynamics Applied to General Aviation/Business Aircraft. Chapter 5, in *Applied Computational Aerodynamics. Progress in Astronautics and Aeronautics*. Vol. 125, P.A. Henne, ed., American Institute of Aeronautics and Astronautics, Inc., Washington DC, 1990.

[26] Martensen, E. Berechnung der Druckverteilung an Gitterprofilen in ebener Potentialstromung mit einer Fredholmschen Integralgleichung. *Archive for Rational Mechanics and Analysis*, Vol. 3, 235-270, 1959.

[27] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M. N., Teller, A.H. and Teller, E., Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, Vol. 21, 1087- 1092, 1958.

[28] Nelder, J. A. and Mead, R., A Simplex Method for Function Minimization. *Computer Journal*, Vol. 7, 308-313, 1965.

[29] Newman, W. M. and Sproull, R. F., Principles of Interactive Computer Graphics, 2nd ed., McGraw-Hill International Book Company, Aukland, 1979.

[30] Obayashi, S. and Tsukahara, T., Comparison of Optimization Algorithms for Aerodynamic Shape Design, *American Institute of Aeronautics and Astronautics* 14^{th} *Applied Aerodynamics Conference*, Paper 96-2394-CP, New Orleans, 1996.

[31] Pfeiffer, N.J., Computational Aerodynamics Applied to General Aviation/Business Aircraft, Chapter 16, in *Applied Computational Aerodynamics, Progress in Astronautics*

and Aeronautics, Vol. 125, P.A. Henne, ed., American Institute of Aeronautics and Astronautics, Inc., Washington DC, 1990.

[32] Pincus, M., A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems, *Operations Research* Vol. 18, 1225-1228, 1970.

[33] Press, W. H., Vetterling, W. T., Teukolsky, S. A., and Flannery, B. P., Numerical Recipes in C: The Art of Scientific Computing, 2nd ed., Cambridge University Press, Cambridge, 1992.

[34] Price, K. and Storn, R., Differential Evolution, *Dr. Dobb's Journal*, Vol. 22, Issue 4, 18-24, April, 1997.

[35] Rechenberg I., Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, Frommann-Holzboog, Stuttgart, 1973.

[36] Sarpkaya, T., Computational Methods with Vortices - The 1988 Freeman Scholar Lecture, *Journal of Fluids Engineering*, Vol. 111, 5-52, March, 1989.

[37] Schwefel, H. P., Evolutionsstrategie und Numerische Optimierung, Dissertation, Technical University of Berlin, 1975a.

[38] Schwefel, H-P., Binäre Optimierung durch Somatische Mutation, Technical Report, Technical University of Berlin and Medical University of Hannover, 1975b.

[39] Smith, A. M. O., The Panel Method, its Original Development, Chapter 1, in *Applied Computational Aerodynamics, Progress in Astronautics and Aeronautics*, Vol. 125, P.A.

Henne, ed., American Institute of Aeronautics and Astronautics, Inc., Washington DC, 1990.

[40] Spendley, W., Hext, G. R. and Himsworth, F. R., Sequential Application of Simplex Designs in Optimization and Evolutionary Operation, *Technometrics*, Vol. 4, No. 4, 441-461, Novermber 1962.

[41] Stewart, W. L. and Glassman, A. J., Blade Design in *Turbine Design and Application, Vol. 2*, National Aeronautics and Space Administration, Washington DC, 1974.

[42] Storn, R. and Price, K., Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, ICSI, ftp.icsi.berkeley.edu, March 1995.

[43] Stratford, B. S., The prediction of Separation of the Turbulent Boundary Layer, *Journal of Fluid Mechanics*, Vol. 5, 1-16, 1959a.

[44] Stratford, B. S., An Experimental Flow with Zero Skin Friction Throughout its Region of Pressure Rise, *Journal of Fluid Mechanics*, Vol. 5, 17-35, 1959b.

[45] Vanderbilt, D. and Louie, S. G., A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables, *Journal of Computational Physics*, Vol. 56, 259-271, 1984.

[46] Vanderplaats, G.N., Hicks, R.N., and Murman, E.M., Applications of Numerical Optimization Technique to Airfoil Design, NASA Ames Research Center, NASA SP-347, Part II, pp 749-768, March, 1975.

[47] Venkataraman, P., A New Procedure for Airfoil Definition, *American Institute of Aeronautics and Astronautics 13th Applied Aerodynamics Conference*, Paper 95-1875-CP, San Diego, 1995a.

[48] Venkataraman, P., Optimum Airfoil Design in Viscous Flows, *American Institute of Aeronautics and Astronautics 13th Applied Aerodynamics Conference*, Paper 95-1876-CP, San Diego, 1995b.

[49] Venkataraman, P., Optimal Airfoil Design, *American Institute of Aeronautics and Astronautics 14th Applied Aerodynamics Conference*, Paper 96-2371-CP, New Orleans, 1996a.

[50] Venkataraman, P. Inverse Airfoil Design using Design Optimization, *American Institute of Aeronautics and Astronautics 14th Applied Aerodynamics Conference*, Paper 96-2503-CP, New Orleans, 1996b.

[51] Walker, G. J., A Family of Surface Velocity Distributions for Axial Compressor Blading and Their Theoretical Performance, *Journal of Engineering for Power*, 229-241, April, 1976.

[52] Wilkinson, D. H., A Numerical Solution of the Analysis and Design Problems for the Flow past one or more Aerofoils or Cascades, *A. R. C. R&M*, No. 3545, 1967.

[53] Wilkinson, D. H., The Analysis and Design of Blade Shape for Radial, Mixed and Axial Turbomachines with Incompressible Flow. *M. E. L. Report No. W/M(3F), English Electric Co.*, Whetstone, Leicester, 1969.

[54] Zweifel, O., The Spacing of Turbo-Machine Blading Especially with Large Angular Deflection, *Brown Boveri Review*, Vol. 32, No. 12, 436-444, December, 1945.

Appendix A

Graphs of Results

This appendix contains plots of the Liebeck shapes that required additional constraints and of the designed Liebeck fan blades with their corresponding thickness and camber curves.

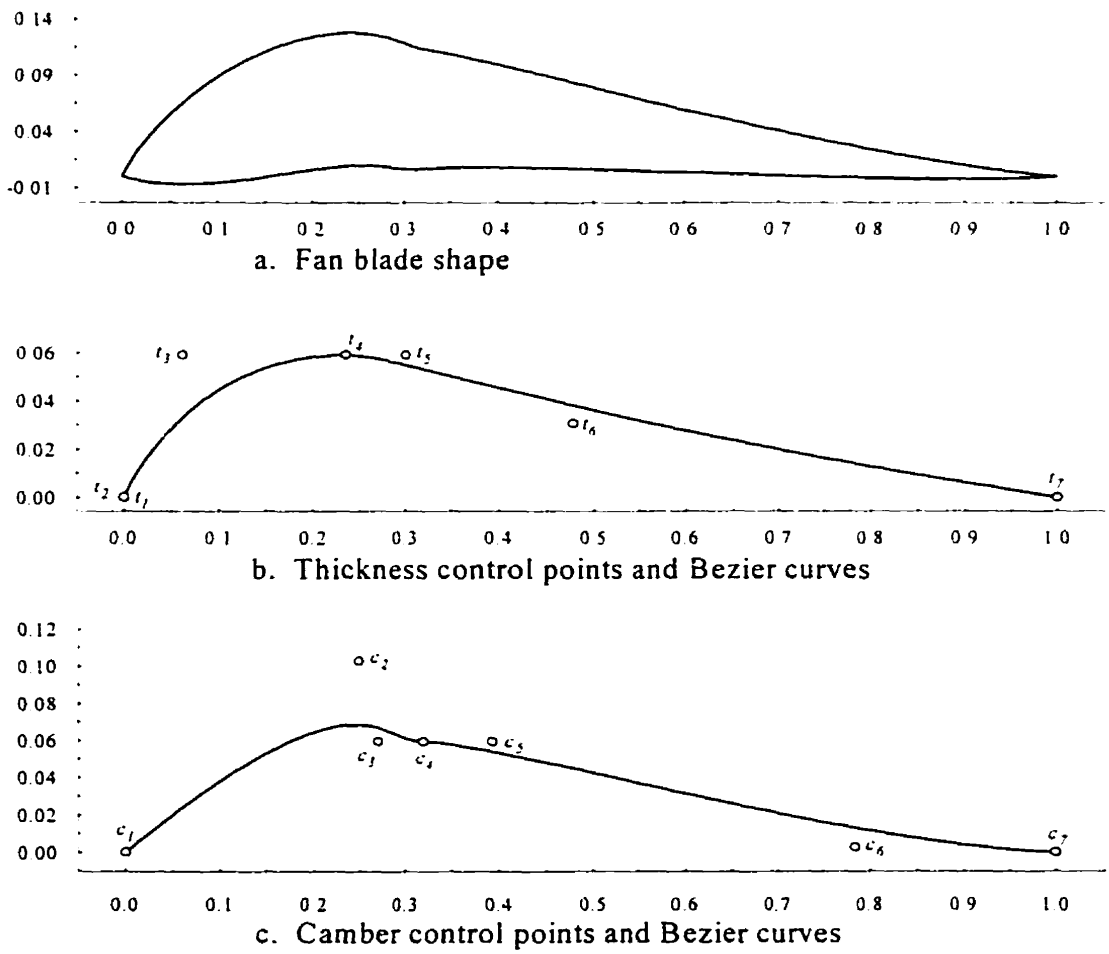
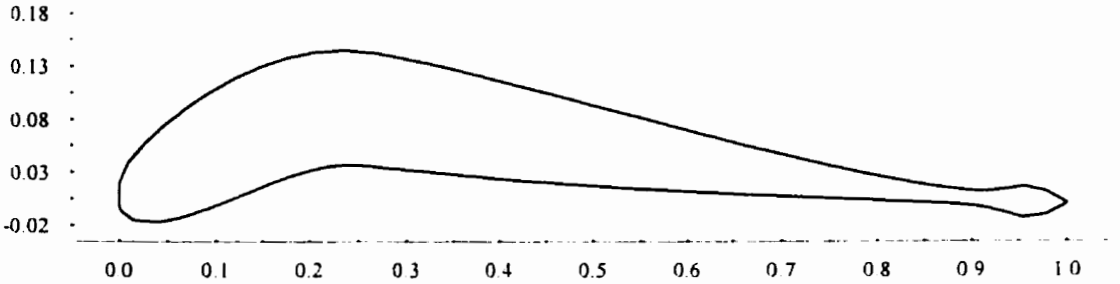
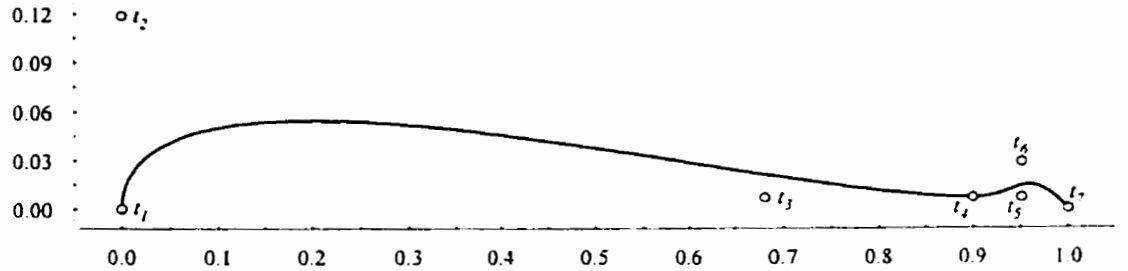


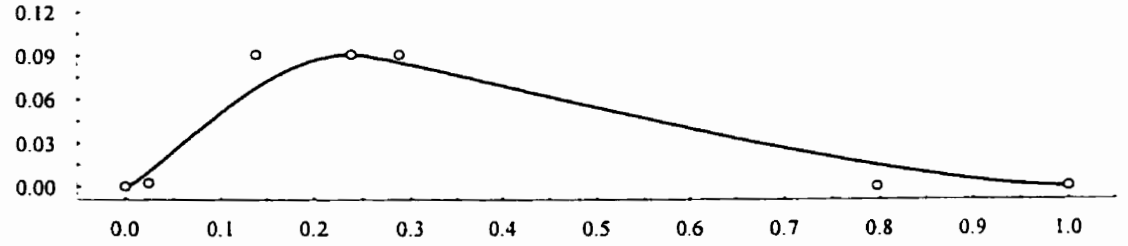
Figure A.1: First Liebeck shape requiring additional constraints.



a. Fan blade shape

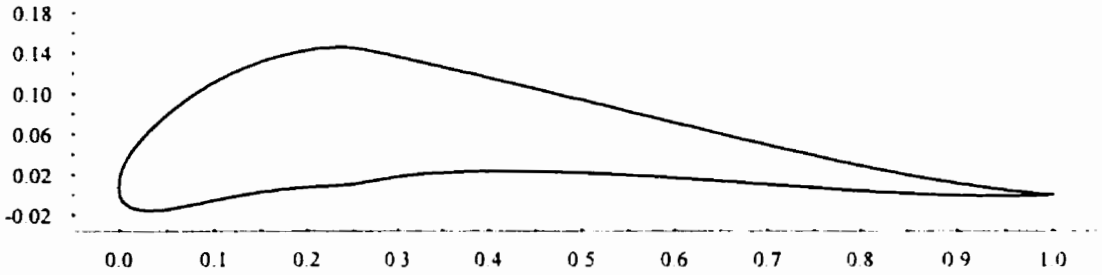


b. Thickness control points and Bezier curves

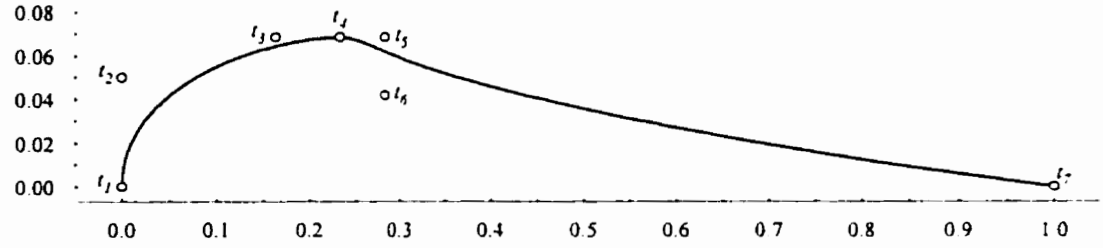


c. Camber control points and Bezier curves

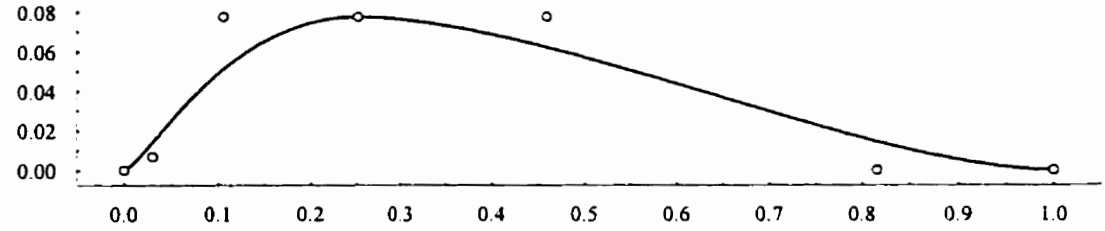
Figure A.2: Second Liebeck shape requiring additional constraints.



a. Fan blade shape

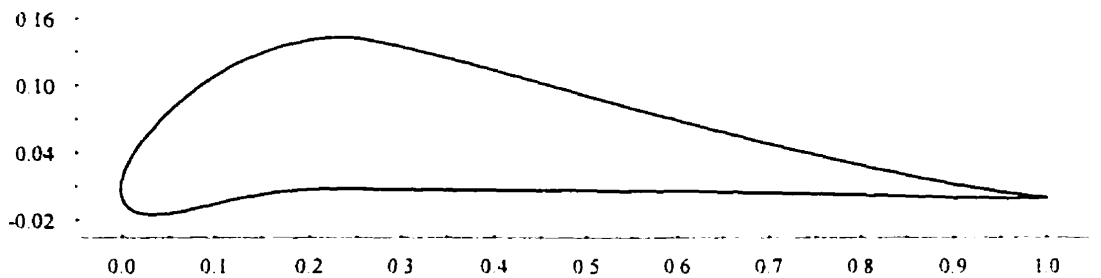


b. Thickness control points and Bezier curves

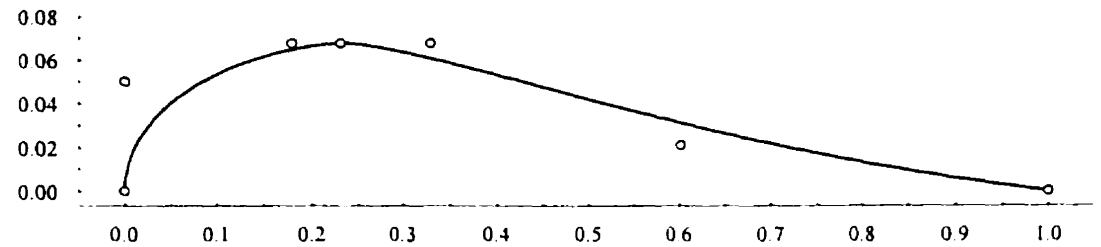


c. Camber control points and Bezier curves

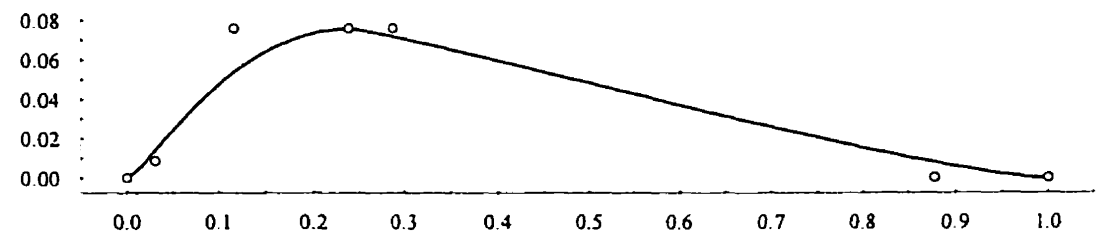
Figure A.3: Third Liebeck shape requiring additional constraints.



a. Fan blade shape

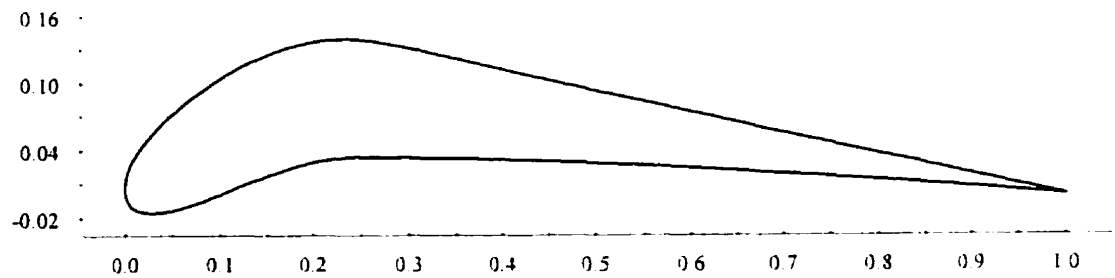


b. Thickness control points and Bezier curves

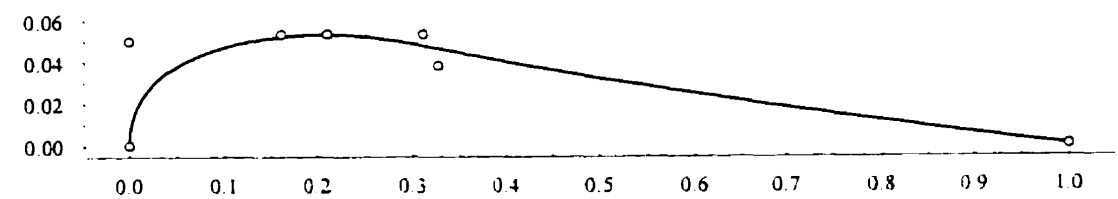


c. Camber control points and Bezier curves

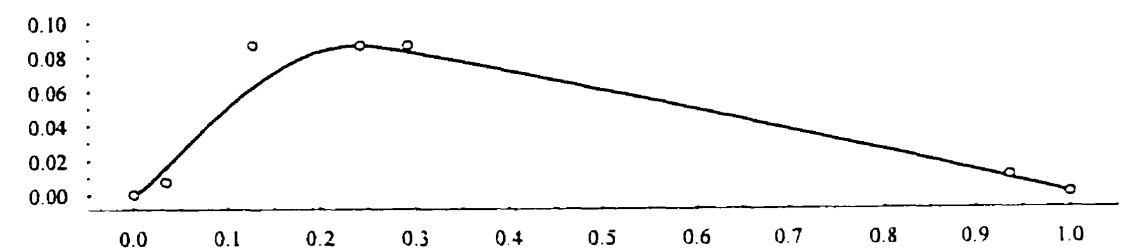
Figure A.4: Bezier curves for Liebeck Design 1 ($\beta_1 = 30$, $\beta_2 = 0$, $t/l = 1.0$, $\lambda = 3.338$).



a. Fan blade shape

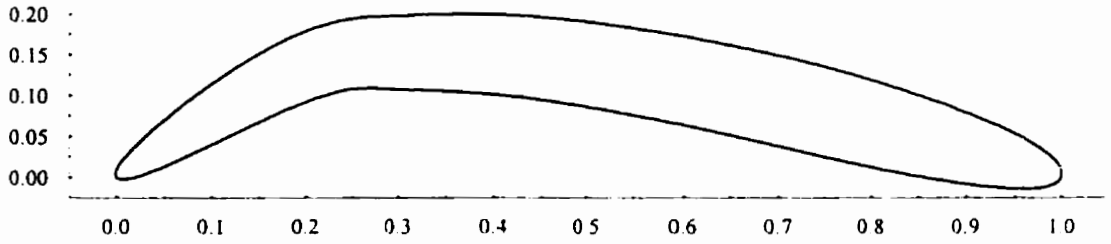


b. Thickness control points and Bezier curves

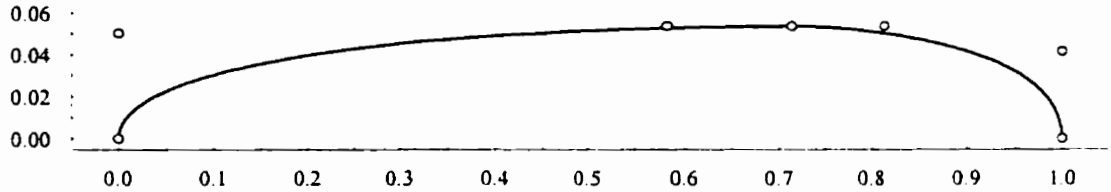


c. Camber control points and Bezier curves

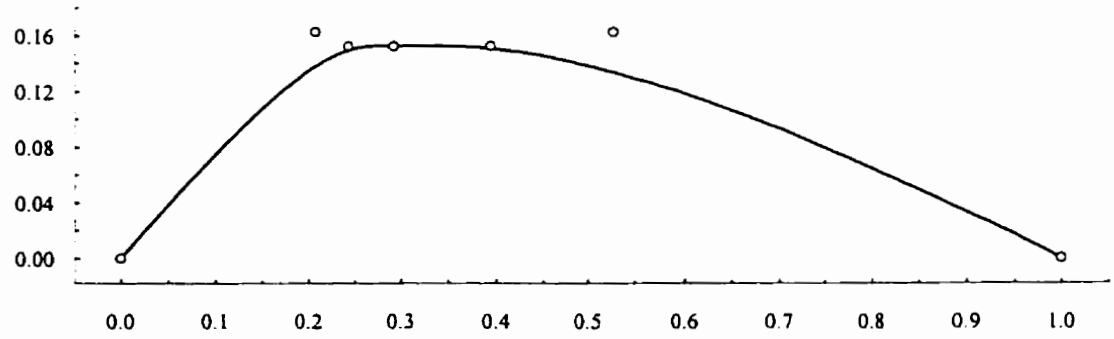
Figure A.5: Bezier curves for Liebeck Design 2 ($\beta_1 = 30$, $\beta_2 = 0$, $t/l = 1.183$, $\lambda = 4.734$).



a. Fan blade shape

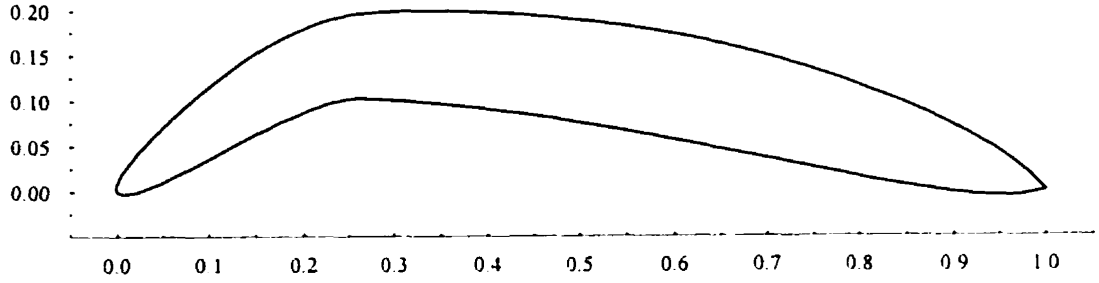


b. Thickness control points and Bezier curves

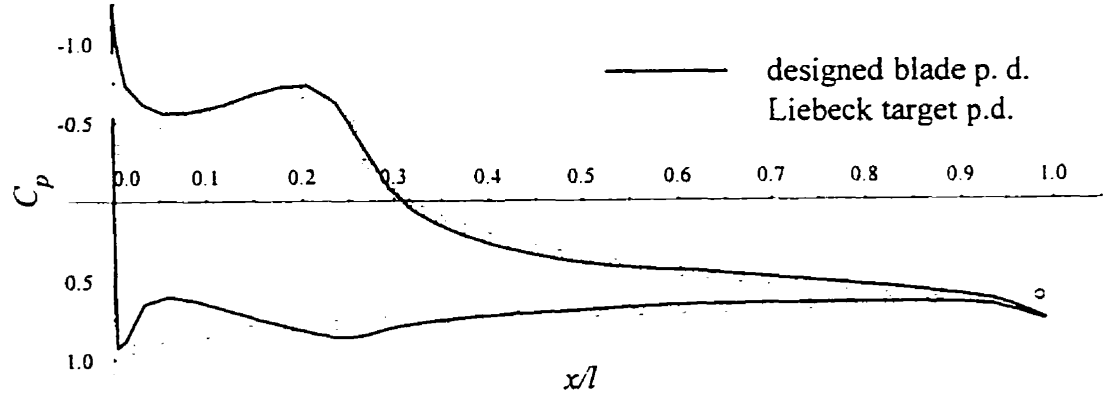


c. Camber control points and Bezier curves

Figure A.6: Bezier curves for Liebeck Design 3a ($\beta_1 = 60$, $\beta_2 = 0$, $t/l = 0.560$, $\lambda = 17.701$).

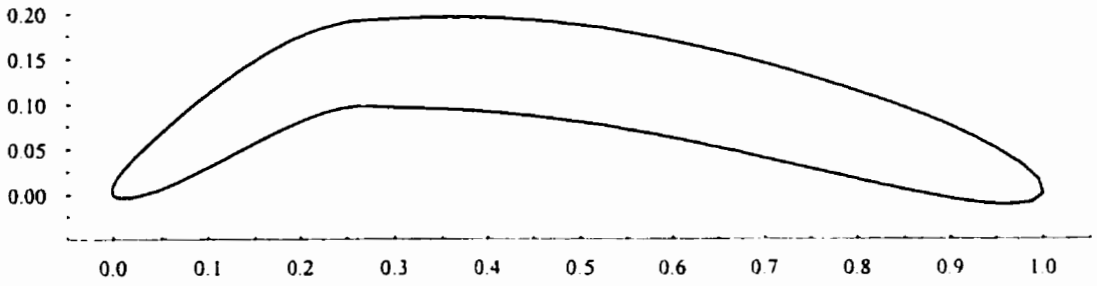


a. Designed fan blade shape

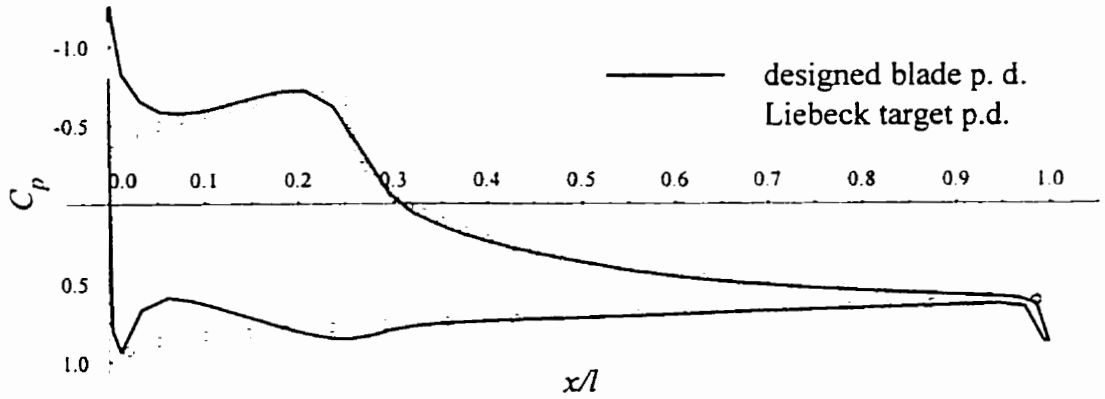


b. Pressure distribution comparison

Figure A.7: Liebeck Design 3b ($\beta_1 = 60$, $\beta_2 = 0$, $t/l = 0.547$, $\lambda = 17.066$).

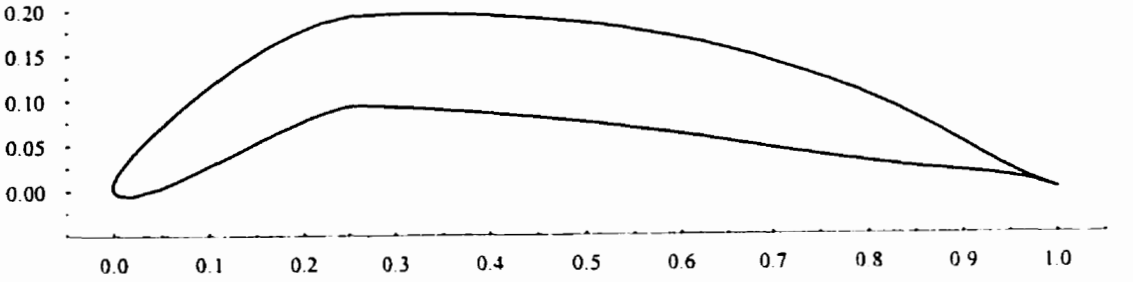


a. Designed fan blade shape

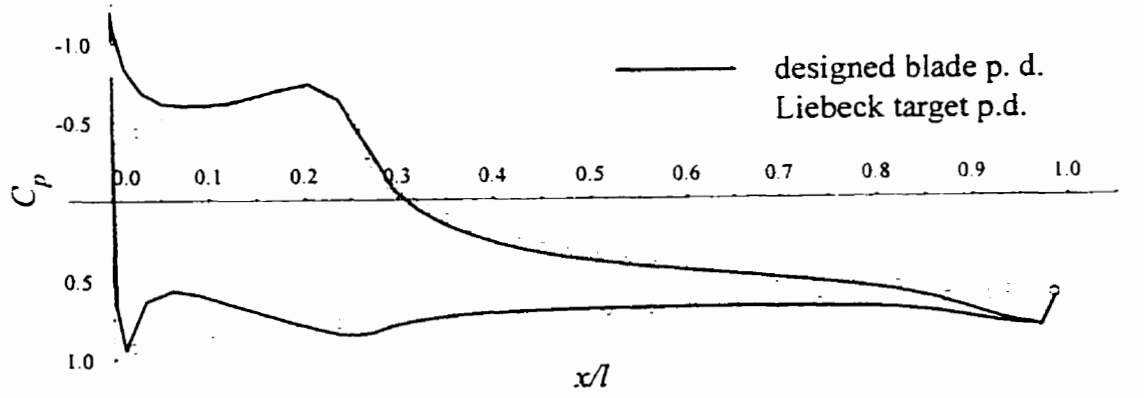


b. Pressure distribution comparison

Figure A.8: Liebeck Design 3c ($\beta_1 = 60$, $\beta_2 = 0$, $t/l = 0.557$, $\lambda = 17.555$).

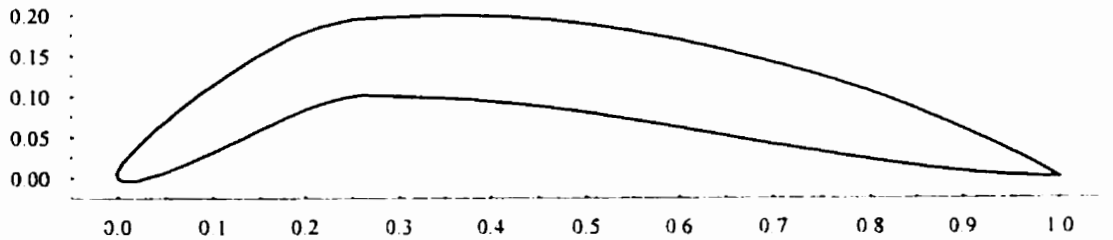


a. Designed fan blade shape

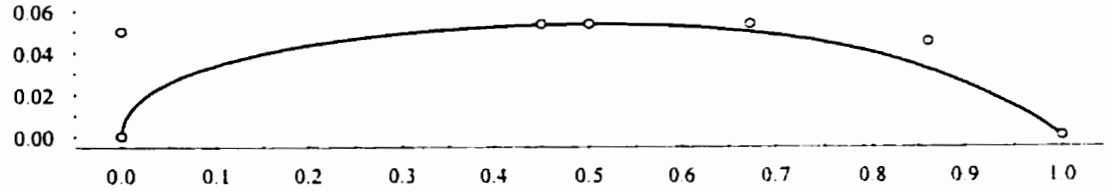


b. Pressure distribution comparison

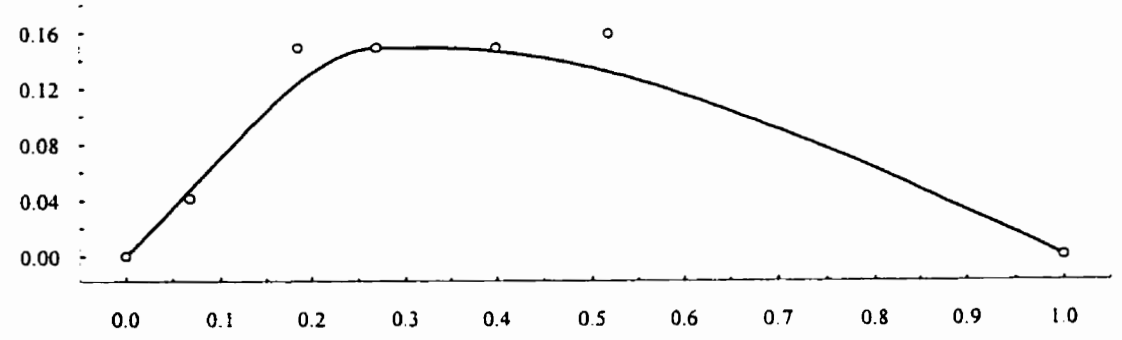
Figure A.9: Liebeck Design 3d ($\beta_1 = 60$, $\beta_2 = 0$, $t/l = 0.546$, $\lambda = 17.013$).



a. Fan blade shape



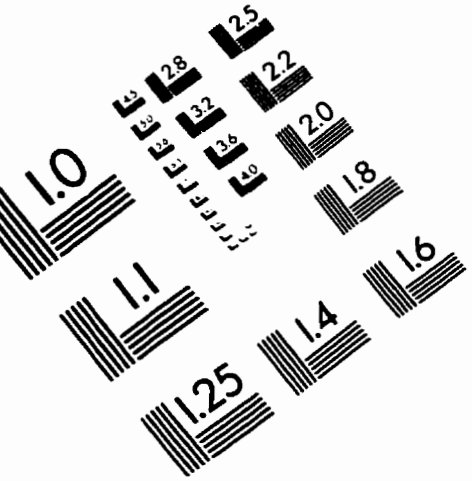
b. Thickness control points and Bezier curves

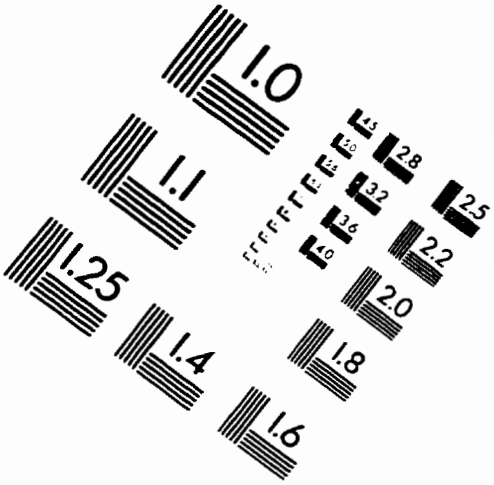


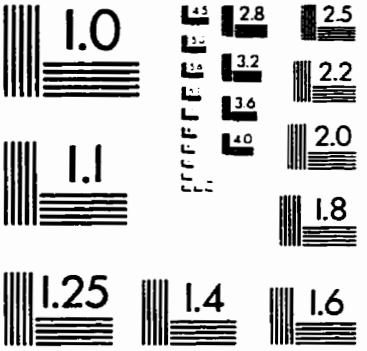
c. Camber control points and Bezier curves

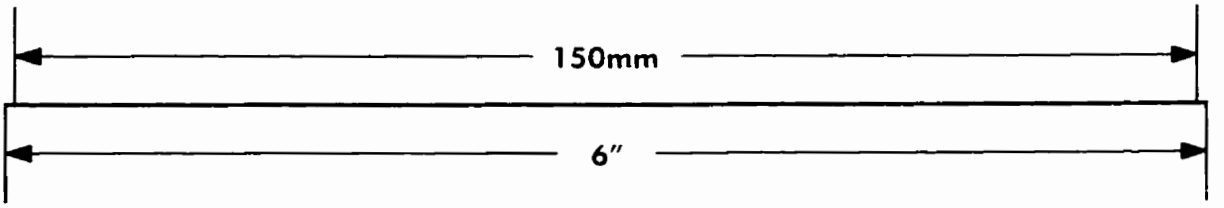
Figure A.10: Bezier curves for Liebeck Design 3e ($\beta_1 = 60, \beta_2 = 0, t/l = 0.553, \lambda = 16.951$).

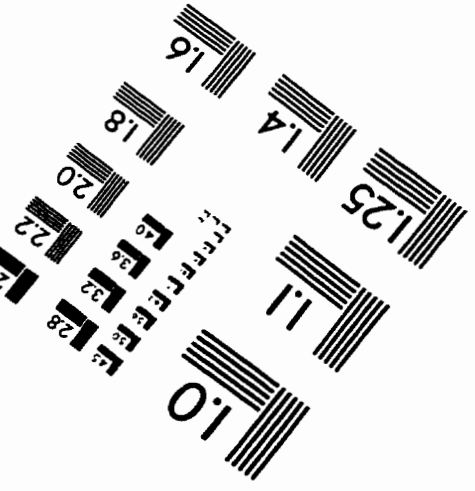
IMAGE EVALUATION
TEST TARGET (QA-3)

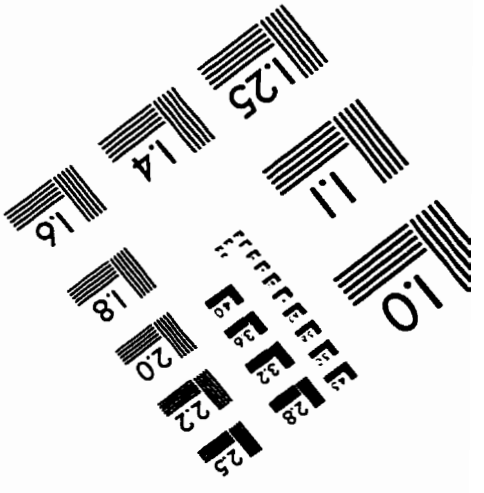












APPLIED IMAGE . Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989